DESARROLLO DE COMPONENTE WEB PARAMETRIZABLE DE CLUSTERING PARA ANÁLISIS DE DATOS

UNIVERSIDAD CATÓLICA DE COLOMBIA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
BOGOTÁ D.C
2018

DESARROLLO DE COMPONENTE WEB PARAMETRIZABLE DE CLUSTERING PARA ANÁLISIS DE DATOS

JOSEPH ALEXANDER RUBIO TAPIAS
CARLOS ANDRÉS ALBA RODRÍGUEZ

TRABAJO DE GRADO PARA OPTAR AL TÍTULO DE INGENIERO DE SISTEMAS

DIRECTOR
DIEGO ALBERTO RINCÓN YÁÑEZ MCSc
INGENIERO DE SISTEMAS

UNIVERSIDAD CATÓLICA DE COLOMBIA
FACULTAD DE INGENIERÍA
PROGRAMA DE INGENIERÍA DE SISTEMAS Y COMPUTACIÓN
BOGOTÁ D.C
2018

NOTA DE ACEPTACIÓN

Aprobado por el comité de grado en cumplimiento de los requisitos exigidos por la Facultad de Ingeniería y la Universidad Católica de Colombia para optar al título de Ingenieros de Sistemas.

_____

Jurado

_____

Diego Alberto Rincón
Director

_____

Revisor Metodológico.

Fecha de Entrega

AGRADECIMIENTOS

Agradecemos a Dios por guiar el camino de nuestra vida personal y profesional.

A nuestros padres por su apoyo, amor y sacrificio incondicional, por confiar y creer en nuestros sueños.

A nuestro Director de Trabajo de Grado, Ing. Diego Alberto Rincón, por haber compartido con nosotros sus conocimientos durante nuestro proceso académico y por orientar la elaboración y producción de este documento.

TABLA DE CONTENIDO

TABLA DE ILUSTRACIONES

LISTA DE TABLAS

TABLA DE ANEXOS

ABSTRACT

The purpose of this project is the development of a Clustering Web Component for Data Analysis to aspire to the degree of Systems Engineer at Universidad Católica de Colombia. This development will allow a set of data to be entered into a web interface and / or an API that will be analyzed by means of clustering techniques, which allow the grouping of data based on their characteristics, and thus yield a result that relates said data. The foregoing, seeking to provide the University with a tool that allows, in a general, practical and freeway, to perform data analysis based on clustering for any research study.

For the elaboration of this project, the phases of the software development process were established: analysis, design, coding, testing, documentation and maintenance. As a theoretical framework for research, concepts related to Big Data, Clustering Methods and Microservices Architecture were investigated. The project was designed with technologies for the development of software such as Java, Spring Boot, REST, R Language, among others. In addition, the methodology of agile development of programming XP (Extreme Programming) was used, whose main objective is to increase productivity when developing a software project.

This component is based on the architecture of microservices and integrates (repays) two types of different technologies that are Java and Language R, in order to guarantee a modular design that expires with the good practices of development of software. For it, also there was in use the methodology XP, and likewise, each of the stages were established of development, which expire with the life cycle of the software.

Key Words: Data Analysis, Clustering Methods, Microservices Architecture.

RESUMEN

Este proyecto tiene como fin el desarrollo de un Componente Web de Clustering para Análisis de Datos para aspirar al grado de Ingeniero de Sistemas de la Universidad Católica Colombia. Este desarrollo permitirá ingresar en una interfaz web y/o una API un conjunto de datos que serán analizados mediante técnicas de clustering, las cuales permiten realizar agrupamiento de datos basados en sus características, y así arrojar un resultado que relaciona dichos datos. Lo anterior, buscando proveer a la Universidad de una herramienta que permita de manera general, práctica y gratuita, realizar análisis de datos basados en clustering para cualquier estudio de investigación.

Para la elaboración de este proyecto se establecieron las fases del proceso de desarrollo de software: análisis, diseño, codificación, pruebas, documentación y mantenimiento. Como marco teórico de investigación se indagaron conceptos relacionados a Big Data, Métodos de Clustering y Arquitectura de Microservicios. El proyecto fue diseñado con tecnologías para el desarrollo de software como Java, Spring Boot, REST, Lenguaje R, entre otras. Además, se utilizó la metodología de desarrollo ágil de programación XP (Extreme Programming), que tiene como objetivo primordial ampliar la productividad al desarrollar un proyecto de software.

Este componente se basa en la arquitectura de microservicios e integra dos tipos de tecnologías diferentes que son Java y Lenguaje R, con el fin de garantizar un diseño modular que cumpla con las buenas prácticas de desarrollo de software. Para ello, también se utilizó la metodología XP, y así mismo, se establecieron cada una de las etapas de desarrollo, las cuales cumplen con el ciclo de vida del software.

Palabras Claves: Análisis de Datos, Métodos de Clustering, Arquitectura de Microservicios.

INTRODUCCIÓN

En la actualidad, el análisis de datos es uno de los campos de acción de las ciencias de la computación con mayor auge, debido a su importancia para generar información a partir de los datos que se analizan o estudian. En este sentido, encontramos la Minería de Datos y el Big Data, siendo este último, un proceso que se encarga de la extracción, análisis, analítica y visualización de resultados de los datos estudiados.

Sin embargo, el análisis de datos se apoya en técnicas y herramientas para realizar la extracción de información de los datos, es así como el clustering, que es una técnica de aprendizaje no supervisado, cobra importancia ya que con esta técnica es posible, mediante diferentes algoritmos, extraer patrones que relacionan los datos, estos patrones se conocen como grupos.

Es evidente que para ejecutar dichas técnicas como el clustering se requieren de soluciones de software que permita ejecutarlas. Hoy en día en el mercado existe una gran cantidad de software que proporciona esta funcionalidad, pero la mayoría son de pago o sus capacidades son limitadas.

La Universidad Católica de Colombia, dentro de sus estudios de investigación, no cuenta con una herramienta que brinde este tipo de técnicas, las cuales son de suma importancia cuando las investigaciones requieren de levantamiento y análisis de datos.

En el transcurrir de este proyecto y apoyados en la ingeniería de software, se desarrollará un **Componente Web Parametrizable de Clustering para Análisis de Datos (CWPCAD),** el cual basado en las últimas tendencias de arquitectura de software como son los microservicios y las tecnologías como Java y lenguaje R, permitirá proveer a la Universidad de una solución robusta, funcional y escalable para realizar análisis de clustering.

1. GENERALIDADES

1.1 ANTECEDENTES

El software como servicio es un modelo de computación distribuida cada vez más demandado por los usuarios, ya que permite obtener acceso a servicios o funcionalidades directamente desde la web. La mayor parte de estos servicios se realiza usando arquitecturas distribuidas y no monolíticas como los microservicios.

Es importante entender que el ecosistema de servicios es variado. En este se encuentran los servicios de Machine Learning, que a su vez, contienen los algoritmos de clustering, los cuales son fundamentales en diversos campos de estudio.

Sin embargo, los servicios de clustering no se ofrecen como alternativas gratuitas y más aún, en el caso de los grandes proveedores de servicios como los son Amazon y su servicio AWS Machine Learning o Microsoft Azure con Machine Learning Services, entre otros, sólo disponen de algunos algoritmos de clustering (particularmente los más conocidos como K-Means).

De acuerdo a lo anterior, es preciso contar con una herramienta de software que, aprovechando los modelos de arquitecturas distribuidas y las ventajas del clustering, permita crear un servicio escalable y portable que ofrezca diferentes algoritmos de clustering de forma parametrizable.

Por otra parte, se presentan a continuación los resultados de una revisión de diferentes herramientas relacionadas al análisis de datos y clustering, con las cuales es evidente que en los últimos años se ha venido trabajando en herramientas y técnicas para el análisis de datos, que son las siguientes:

1.1.1 RapidMiner:

Es una herramienta informática para el análisis y minería de datos, basada en operaciones que descubren patrones en grandes colecciones de texto. También, es una plataforma analítica que integra el aprendizaje de máquina y despliega un modelo predictivo utilizado por científicos de datos. Esta plataforma está conformada por varias bibliotecas de ciencias de datos y algoritmos de aprendizaje de máquina y es usada y altamente probada en investigación, educación, capacitación, creación rápida de prototipos y en aplicaciones empresariales.

RapidMiner cuenta con varios componentes o versiones, entre los cuales se encuentran: RapidMiner Studio, RapidMiner Auto Model, RapidMiner Turbo Prep,

Servidor RapidMiner, RapidMiner Radoop. A su vez, implementa todos los operadores de data mining, modelos predictivos, modelos descriptivos, transformación de datos, series de tiempo, etc.

En el 2015, al finalizar la encuesta anual del portal Internacional de Minería de datos KDnuggets, RapidMiner se ubicó en el segundo lugar como la herramienta de Data Mining más utilizada por expertos ("RapidMiner | Sistemas de Minería de Datos | Software de Minería de Datos," n.d.).

1.1.2 Weka

Contiene una colección de herramientas de visualización y algoritmos para análisis de datos y modelado predictivo, unidos a una interfaz gráfica de usuario para acceder fácilmente a sus funcionalidades, especificando parámetros de entrada y salida (Garner, n.d.). Esta herramienta fue desarrollada inicialmente en 1997, con el objetivo de analizar datos relacionados con la agricultura, la versión original de Weka fue un frontend, para modelar algoritmos implementados en otros lenguajes de programación; actualmente tiene como finalidad facilitar una amplia gama de algoritmos de Machine Learning y procesamiento de datos y es usada en diferentes áreas. Para su fácil manejo la herramienta ofrece una API simple, mecanismos de plugin e instalaciones que automatizan la integración de nuevos algoritmos de aprendizaje e incluye algoritmos de regresión, clasificación, agrupación, minería de reglas de asociación y selección de atributos (Hall et al., n.d.) . Se puede decir que Weka está desarrollada en tres grandes categorías:

- **Procesamiento de Conjuntos de Datos:** Este módulo se encarga de la extracción de información desde un conjunto de datos, los cuales son divididos en conjuntos de prueba y de entrenamiento, y posteriormente son transformados de tal manera que permitan trabajar con un esquema de aprendizaje automático.

- **Esquemas de Aprendizaje Automático**: Es un conjunto de reglas, basado en algoritmos de aprendizaje automático, que permite realizar operaciones que generan una salida.

- **Procesamiento de Salida:** Este evalúa un conjunto de reglas contra una prueba y genera un archivo o una visualización de la salida en una ventana manejable por el usuario.

1.1.3 JHepWork

Es un marco de trabajo gratuito de análisis de datos para científicos, ingenieros y estudiantes, desarrollado en lenguaje de programación Java. JHepWork permite establecer un ambiente de análisis de datos sin costo o código abierto con una interfaz de usuario intuitiva y se convierte en una herramienta competitiva ante el software comercial. Está basado en bibliotecas numéricas científicas implementadas en Java para funciones matemáticas, números aleatorios, análisis estadístico, ajuste de curvas de regresión y otras actividades de minería de datos.

1.1.4 KNIME (Konstanz Information Miner)

KNIME es un software que integra varios componentes para el Machine Learning y la minería de datos a través de su concepto de canalización de datos modular. La interfaz gráfica del usuario y el uso de JDBC permiten el ensamblaje de nodos que combinan diferentes fuentes de datos, incluido el preprocesamiento (ETL: Extracción, Transformación, Carga), para el modelado, análisis de datos y visualización sin o con sólo una programación mínima. KNIME está enfocado en proporcionar funcionalidades para la creación de informes profesionales, análisis estadísticos, químicos, informática y muy recientemente, análisis de imagen de alto rendimiento / alto contenido (HCS / HCA) y secuencias de comandos en lenguajes como Perl, Python, Matlab, R y Java (Jagla, Wiswedel, & Coppée, 2011).

1.1.5 Orange

Es un software de programación visual fundamentado en módulos para la visualización de datos, aprendizaje automático, extracción de datos y análisis de datos. Los componentes de Orange se denominan widgets y van desde la simple visualización de datos, la selección de subconjuntos y el preprocesamiento, hasta la evaluación empírica de algoritmos de aprendizaje y el modelado predictivo.

La programación visual se realiza por medio de una interfaz en la que se crean flujos de trabajo mediante la vinculación de widgets ya diseñados por el usuario, los usuarios expertos pueden usar Orange como una biblioteca de Python para la administración de datos y la alteración de dichos widgets.

Orange facilita el análisis de datos simples con visualización de datos inteligente y permite distribuciones estadísticas, diagramas de caja y diagramas de dispersión, además profundiza con árboles de decisión, agrupación jerárquica, mapas de calor, MDS y proyecciones lineales. Incluso sus datos multidimensionales pueden llegar a ser sensibles en 2D, especialmente con la clasificación y selección de atributos inteligentes.

1.2 PLANTEAMIENTO DEL PROBLEMA

En general, en la mayoría de los procesos de investigación (de cualquier índole o tema) se realizan levantamientos o recolección de datos con los que se pretende extraer la información que ellos contienen, de manera que se puedan sacar conclusiones que generen valor a la investigación y que sean de utilidad para la misma, esto se realiza mediante técnicas de análisis de datos.

Dentro del análisis de datos, a menudo se requiere clasificar la información de acuerdo con sus características y similitudes, para ello, se usan técnicas de clasificación primordialmente apoyadas en Machine Learning. Una de ellas es denominada ‘análisis de clustering’ que permite, mediante diferentes algoritmos, extraer patrones en los datos que los relacionan, es decir grupos.

Actualmente, existen diferentes herramientas en el mercado que ofrecen este tipo de análisis pero en su mayoría son herramientas de pago, es decir que requieren de algún tipo de licenciamiento y las que son de uso gratuito u ‘open source’ contiene un conjunto limitado de funciones. Por otra parte, son soluciones poco escalables y difíciles de integrar con otros sistemas, y aquellas que lo permiten como las que se basan en ‘cloud’ cobran mucho más por esta característica.

La Universidad Católica de Colombia demandó la necesidad de tener una herramienta que permita a sus académicos e investigadores realizar análisis clustering para sus diferentes estudios de investigación, y además, de manera práctica y económica. Por ello, se ha venido trabajando en desarrollar una herramienta que pueda ser utilizada sin necesidad de adquirir software de terceros que involucren costos en la extracción de información y que permita integrarse con otras aplicaciones. Por lo anterior, surge el siguiente interrogante, ¿es posible desarrollar un componente de software que permita realizar análisis tipo clustering, que pueda ser parametrizable y, que, además, pueda integrarse con otras herramientas de software?

1.3 DESCRIPCIÓN DEL PROBLEMA

El ‘análisis de clustering’ es una herramienta importante en el análisis de datos ya que permite encontrar y agrupar información dentro de un conjunto de datos. Dentro de sus principales características está el poder agrupar los datos basado en la similitud de sus variables, sin que estas sean establecidas o etiquetadas, con el fin de encontrar relaciones entre ellos. Un caso práctico que se puede mencionar en donde el clustering es muy importante, es en los procesos de clasificación de información, más aún cuando se tratan de grandes conjuntos de datos.

Adicionalmente, dentro del contexto del análisis de datos, el clustering es complementario y sirve como entrada para otras herramientas como el ‘análisis visual’, el cual permite interpretar los datos numéricos en gráficos.

Actualmente, la Universidad Católica de Colombia no cuenta con una herramienta que permita realizar análisis de clustering para los diferentes estudios de investigación del ámbito académico, además no puede asumir costos adicionales para llevar a cabo estos análisis. Por consiguiente, se pierde la posibilidad de extraer la información que sólo con el análisis de clustering se puede lograr, privándose así de generar nuevo conocimiento en los estudios de investigación donde se involucra la recolección de datos.

Para ello, es necesario conocer e investigar sobre los principales algoritmos de clustering que se usan para llevar a cabo el análisis de datos, así como conocer las soluciones tecnológicas que los apoyan y que permiten desarrollarlos. Adicionalmente, y con el fin de permitir la escalabilidad y la integración con otros componentes o herramientas de software, es importante determinar cuáles son las últimas tendencias en el desarrollo modular de soluciones de software.

Por lo tanto, se pretende proveer a la Universidad de una herramienta o componente de software ‘Ad Hoc’ para realizar análisis de clustering, la cual pueda ser integrada con otros sistemas de información. Realizar este componente permitirá no sólo el aprovechamiento del clustering, sino también su despliegue en cualquier ambiente, y además, mediante patrones de diseño de software y el uso de las últimas tecnologías, podrá ser escalable de ser necesario.

1.4 DELIMITACIÓN

En los siguientes numerales se señalan aspectos importantes que afectan de manera positiva este desarrollo.

1.4.1 Alcance

Diseño y desarrollo del Componente Web Parametrizable de Clustering para Análisis de Datos (CWPCAD).

Este componente expondrá sus funcionalidades a través de tecnologías web y permitirá ejecutar algoritmos de clustering, basado en una entrada de datos (previamente definida por el usuario) y que retornará un resultado.

Permitirá lo siguiente:

- Usar arquitectura de microservicios.
- Exponer un API con los métodos de clustering.
- Disponer de una interfaz web para el cargue de datos por parte del usuario.
- Parametrizar cada algoritmo de clustering.

1.4.2 Tiempo

El desarrollo de este proyecto se llevará a cabo en menos de 1 año (6 meses aproximadamente).

1.4.3 Limitaciones

- El desarrollo de la herramienta se ve limitado por tiempo definido.
- La implementación del prototipo se concretará en el segundo semestre del 2018.
- El tiempo de desarrollo está limitado por el periodo académico.

2. OBJETIVOS DEL PROYECTO

2.1 OBJETIVO GENERAL

Desarrollar un componente de software basado en microservicios que permita realizar análisis de datos mediante técnicas de clustering (agrupamiento de datos) y que mediante una interfaz (Web y API) permita ejecutarlos y obtener los resultados.

2.2 OBJETIVOS ESPECÍFICOS

- Realizar un estado del arte de técnicas asociadas al clustering que puedan ser usadas en el análisis de datos.
- Realizar el levantamiento de requerimientos.
- Desarrollar las cuatro técnicas de clustering con las cuales se analizarán los datos.
- Ejecutar las pruebas del componente de clustering parametrizable con el cual se analizarán datos.

3. MARCO REFERENCIAL

3.1 ESTADO DEL ARTE

En los últimos años, el interés por los datos ha aumentado significativamente y con esto ha venido una evolución en los medios sobre ciencia de datos y Big Data, lo que puede causar un gran desconcierto teniendo en cuenta que existen definiciones ambiguas o sin sentido de estos dos amplios términos.

Según la publicación de 2009, de Nathan Yau, “Rise of the Data Scientist” se identifican tres grandes habilidades de los fanáticos de los datos, por ende, los científicos de datos deben contar con estas características:

- Estadísticas: Análisis tradicional en el que se está acostumbrado a pensar.
- Munging de datos - análisis, raspado y formato de datos.
- Visualización - gráficas, herramientas, etc.

A menudo, una descripción o definición de científicos de datos se refiere a un estadístico híbrido, un ingeniero de software y un científico social. Esto tenía sentido en el contexto de las empresas donde el producto era un producto social y aún tiene sentido cuando tratamos con el comportamiento humano o del usuario (Cathy O’Neil, 2013). Es casi imposible encontrar un científico de datos con todas las cualidades mencionadas, pues puede tener conocimiento en varias áreas, pero siempre será experto en sólo una de ellas, por esta razón es mejor hablar de equipos de ciencia de datos, en los cuales se pueda incluir varios perfiles de científicos de datos (Machine Learning, matemáticas, estadística, etc.), que sean capaces de resolver diferentes tipos de problemas, como se evidencia en la figura (Ver Ilustración 1).

Ilustración 1 - Perfiles del Equipo de Ciencia de Datos

Fuente: Rachel Schutt & Cathy O`Neil Doing Data Science_ Straight Talk from the Frontline-O'Reilly Media (2013)

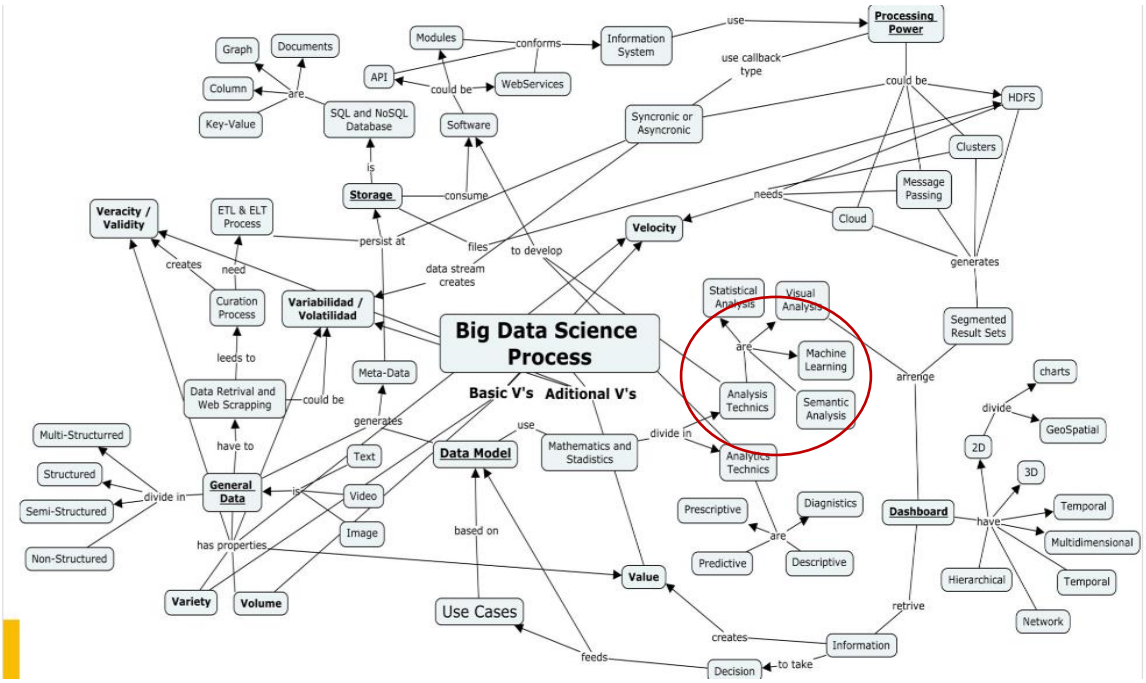
No se puede dejar de lado el término Big Data, el cual hace parte de la ciencia de datos. Éste detalla, maneja y analiza un gran volumen de datos (estructurados y no estructurados) con el cual se suponen mejores decisiones frente a un problema seleccionado. El concepto cobró impulso y ahora es muy popular y conocido por las tres V's (Doug Laney, 2001), las cuales se refieren a Volumen, Velocidad y Variedad.

- **Volumen:** Las organizaciones recopilan datos sin importar la fuente, como lo son transacciones comerciales, redes sociales e información de sensores o datos que se transmiten entre máquinas, usualmente estos datos son almacenados.

- **Velocidad**: Los datos se comparten, tratan y analizan a altas velocidades y se deben conocer de manera oportuna, la medición inteligente crea la necesidad de distribuir cantidades de datos casi en tiempo real.

- **Variedad:** Los datos se tratan sin importar los formatos en los que se recolecten, es posible trabajar con datos numéricos estructurados los cuales usualmente son almacenados en bases de datos relacionales, hasta datos no estructurados como video, audio, correo electrónico, etc.

Ilustración 2 - Proceso Big Data



Fuente: Diego Rincón (2016)

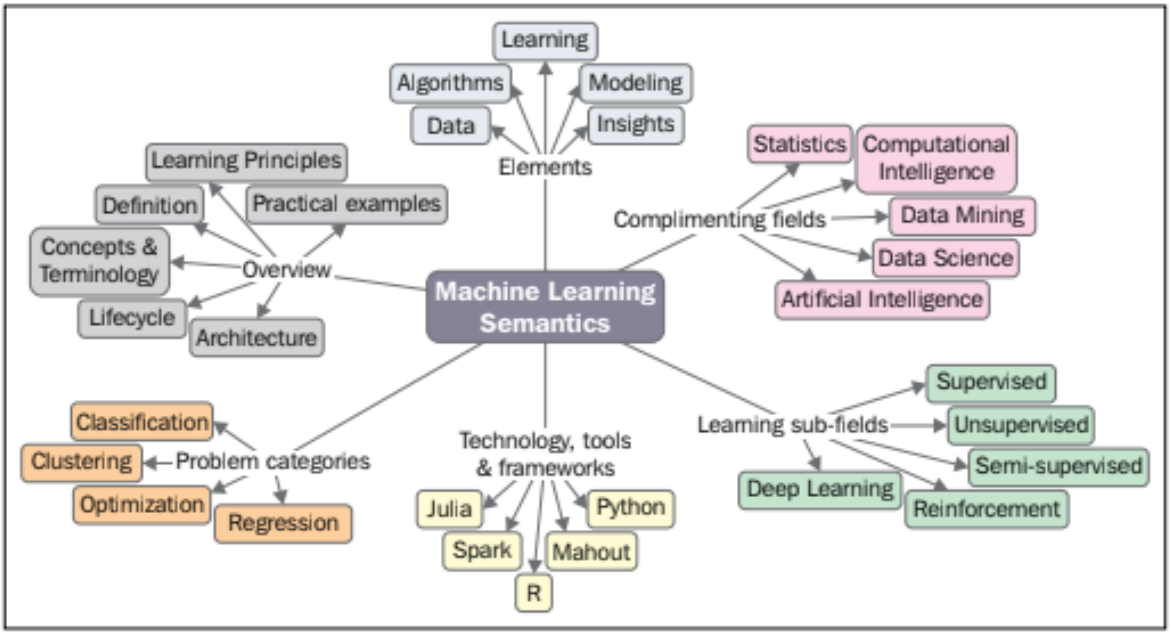
Como se evidencia en la figura (Ver Ilustración 2), Machine Learning hace parte del gran "macroproceso" de Big Data como una técnica de análisis, pero realmente, ¿qué es Machine Learning?, se dice que un programa de computadora aprende de la 'experiencia E' con respecto a alguna clase de las 'tareas T' y la 'medida de rendimiento P', si su desempeño en tareas en T, como medido por P, mejora con la experiencia E (Mitchell, 1997).

La mayoría de las definiciones son relacionadas con patrones, algoritmos, estadística y matemática, pero se puede simplificar que Machine Learning básicamente es un mecanismo para la búsqueda de patrones y la construcción de inteligencia en una máquina para poder aprender, lo que implica que podrá hacerlo mejor en el futuro a partir de su propia experiencia.

El estudio de cómo se percibe el ambiente, cómo se aprende a discriminar el comportamiento y cómo se es capaz de tomar decisiones razonables sobre categorizar el comportamiento (Gollapudi, n.d.), es ciertamente propio de los seres humanos, pero de esto también trata Machine Learning, de que los algoritmos sean desarrollados para que las máquinas den sentido a los datos de la misma manera que lo hacen los humanos.

Machine Learning es un campo que reúne inteligencia artificial, estadística y minería de datos, para que con algoritmos se solucionen problemas de clustering, clasificación, optimización y regresión (Ver Ilustración 3).

Ilustración 3 - Concepto de Machine Learning



Fuente: Gollapudi, Sunila, Practical Machine Learning

El ser humano desde siempre ha sentido la necesidad de clasificar cosas similares en categorías y asignarles etiquetas para su reconocimiento, además de ser una acción humana básica, la clasificación también es indispensable para la mayoría de las ramas de la ciencia. Muchas definiciones de clustering apuntan a la agrupación, pero no para todas las situaciones, muchos autores, por ejemplo, Cormack (1971) y Gordon (1999), intentan definir qué es un clúster en términos de cohesión interna - homogeneidad externa, aislamiento – separación (Everitt, Landau, Leese, & Stahl, n.d.).

Desde su descubrimiento hasta la implementación del 'Data Mining' los algoritmos de clustering han tomado un papel determinante y fundamental a la hora de poder realizar agrupamiento e identificación de patrones y datos, basado en técnicas de aprendizaje no supervisado, en las cuales los datos de entrada no tienen un patrón o clasificación previa establecida y donde el algoritmo por sí solo determina los grupos encontrados de acuerdo a sus características.

Existen dos tipos de clustering (KAUSHIK, 2016):

- **Clustering duro:** En el cual para cada dato existe, si y sólo si, un clúster o grupo asociado.

- **Clustering suave:** En donde para cada dato existe una probabilidad de ocurrencia en cada clúster identificado.

Esta sección conlleva a una categorización de los algoritmos de clustering y se enfocará principalmente en los que se desarrolla el Componente Web Parametrizable de Clustering para Análisis de Datos, los algoritmos serán divididos en 5 categorías (Timmerman, 2012) como sigue a continuación:

- Clustering por particiones.

- Clustering jerárquico.

- Clustering basado en densidad.

- Clustering basado en cuadrícula.

- Clustering basado en modelos.

Clustering por particiones: El objetivo del clustering particional es alcanzar una partición de los datos u objetos en grupos, los cuales se denominan clústeres, de tal manera que todos los datos pertenezcan a alguno de los k clústeres posibles y que por otro lado los clústeres sean distintos, en otras palabras, si se tiene una gran cantidad de datos en un conjunto (clústeres), este se divide en k número de clústeres, cada dato será enviado a un clúster creado dependiendo de su relación. Si denotamos por O = {O1, . . . , ON} al conjunto de N objetos, se trata de dividir O en k grupos o clústeres, Cl1, . . . Clk de tal forma que (Moujahid, n.d.):

$$\mathrm{U}_{\mathrm{j}}^{\mathrm{k}} = 1\mathrm{Clj} = 0$$

ClJ n ClI = 0 para i ≠ j

En el Algoritmo K-means, por ejemplo, un centro es el promedio de todos los puntos y coordenadas que representan la aritmética media. En el algoritmo K-medoids, los objetos que están cerca del centro representan los racimos. Hay muchos otros algoritmos de partición tales como k-Means, PAM/CLARA/CLARANS, BFR.

Clustering jerárquico: Se organizan los datos jerárquicamente según el medio de proximidad. Las proximidades se obtienen por el intermedio de nodos, un dendrograma (diagrama en forma de árbol) representa los conjuntos de datos, donde los datos individuales se presentan por nodos foliares. El grupo inicial de datos se divide gradualmente en varios grupos como la jerarquía continúa. Los métodos de agrupamiento jerárquico pueden ser aglomerante (de abajo hacia arriba) o divisivo (de arriba a abajo). Un agrupamiento aglomerado comienza con un objeto para cada grupo y recursivamente fusiona dos o más de los grupos más apropiados. Se inicia un agrupamiento divisivo con el conjunto de datos como un grupo y se divide de forma recursiva al clúster más apropiado (Timmerman, 2012). En esta clasificación se encuentran los métodos Diana/Agnes, BIRCH, CURE, Chameleon, ROCK que son algunos de los algoritmos conocidos de esta categoría.

Clustering basado en densidad: Aquí los objetos de datos están separados en función de sus regiones de densidad, conectividad y frontera. Están estrechamente relacionados con los vecinos más cercanos al punto. Un clúster, definido como un componente densamente conectado, crece en cualquier dirección a la que conduce la densidad. Por lo tanto, los algoritmos basados en densidad son capaces de descubrir racimos de formas arbitrarias. Además, esto proporciona una protección natural contra los valores atípicos. Así la densidad global de un punto se analiza para determinar las funciones de conjuntos de datos que influyen en un determinado punto de datos. DBSCAN, OPTICS, DBCLASD y DENCLUE son los algoritmos que utilizan un método de este tipo para filtrar el ruido (outliers) y descubrir racimos de forma arbitraria (Timmerman, 2012).

Clustering basado en cuadrícula: Este algoritmo divide el espacio de los objetos de datos en un número finito de celdas o rejillas, una vez se tenga una estructura de cuadrícula continua con todas las operaciones de agrupamiento en clúster (Lin, Chang, Chueh, Chen, & Hao, 2008). Si se tienen calculados los valores estadísticos de las grillas, los datos asignados a cada cuadrícula hacen técnicas de agrupamiento basadas en una cuadrícula independiente para recopilar los datos estadísticos regionales, y luego realizar el agrupamiento en la cuadrícula, la principal ventaja de este enfoque es su tiempo de procesamiento rápido, puesto que divide el conjunto de datos. El rendimiento de una red basada en el presente método depende del tamaño de la cuadrícula, que generalmente es mucho menos que el tamaño de la base de datos, es de aclarar que para distribuciones de datos altamente irregulares, la red puede no ser suficiente para obtener la agrupación

con la calidad esperada o cumplir con el requisito de tiempo. Wave-Cluster, STING y DGD son ejemplos típicos de esta categoría.

Clustering basado en modelos: Este método optimiza el ajuste entre los datos (predefinidos) matemáticos y se basa en el supuesto de que los datos son generados por una mezcla de distribuciones de probabilidad subyacentes, además determina de una forma automática el número de clústeres basados en estadísticas, teniendo en cuenta el ruido (valores atípicos) y por lo tanto dando un método de agrupamiento robusto (Timmerman, 2012).

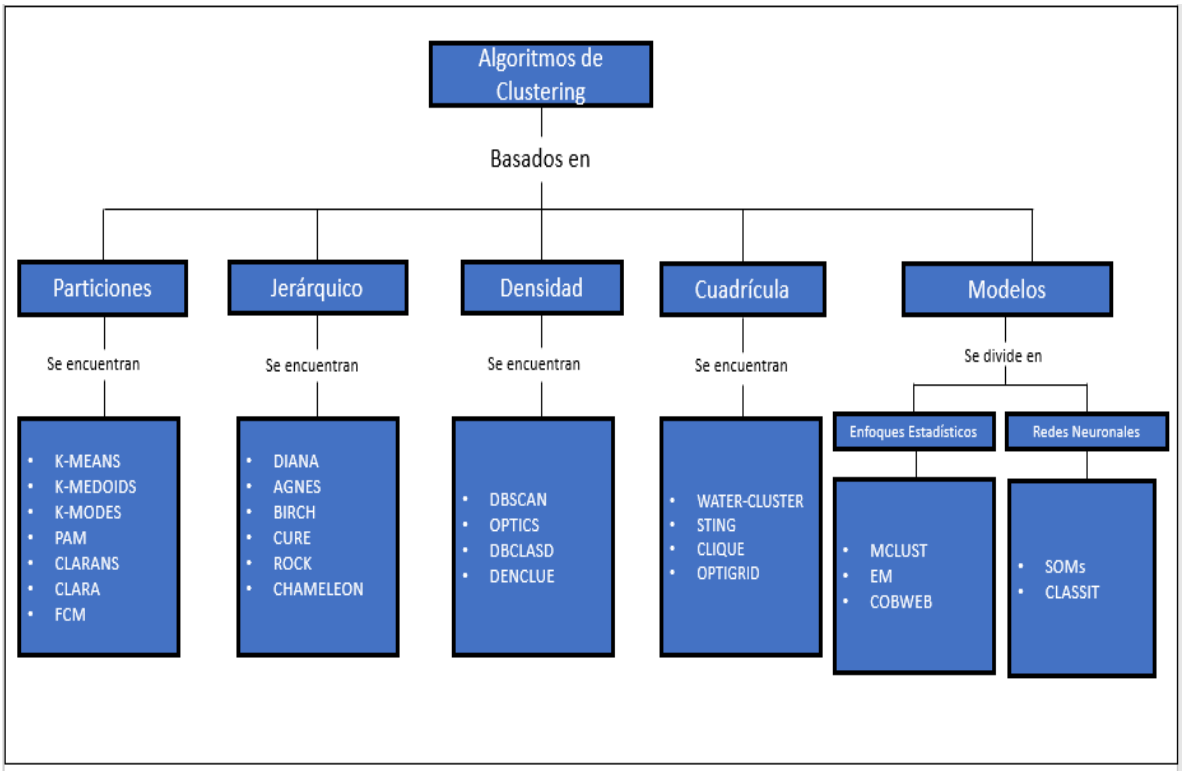
Hay dos grandes enfoques que se basan en el método basado en modelos:

- **Enfoques estadísticos:** MCLUST, este es quizá el algoritmo basado en modelos más conocido, pero existen otros algoritmos con gran eficiencia, como EM (que utiliza un modelo de densidad de mezcla), agrupamiento conceptual (como COBWEB), estos utilizan medidas de probabilidad en la determinación de la conceptos o agrupaciones.

- **Redes neuronales:** En este enfoque se encuentran mapas auto-organizados utilizando un conjunto de entrada / salida, donde cada conexión tiene un peso asociado, para adaptarse mejor a los datos, esto les permite normalizarse. Las redes neuronales son arquitecturas de procesamiento inherentemente paralelas y distribuidas que procesan vectores numéricos y requieren patrones de objetos para ser representados por sólo características cuantitativas. La red neuronal como enfoque de agrupación tiende a representar cada agrupación como ejemplar, un ejemplar actúa como prototipo de clúster y no necesariamente tiene que corresponder a un objeto particular. Se pueden asignar nuevos objetos al clúster cuyo ejemplar es el más similar, basado en alguna medida de distancia (Timmerman, 2012).

El análisis de clústeres consiste esencialmente en descubrir grupos de datos y en agruparlos, los métodos no deben confundirse con la discriminación y los métodos de asignación (en el mundo de la inteligencia artificial se utiliza el término aprendizaje supervisado), donde los grupos son conocidos a priori y el objetivo del análisis es construir las reglas para clasificar nuevos individuos en uno u otro de los grupos conocidos (Hand, 1981).

Ilustración 4 - Taxonomía Algoritmos Clustering

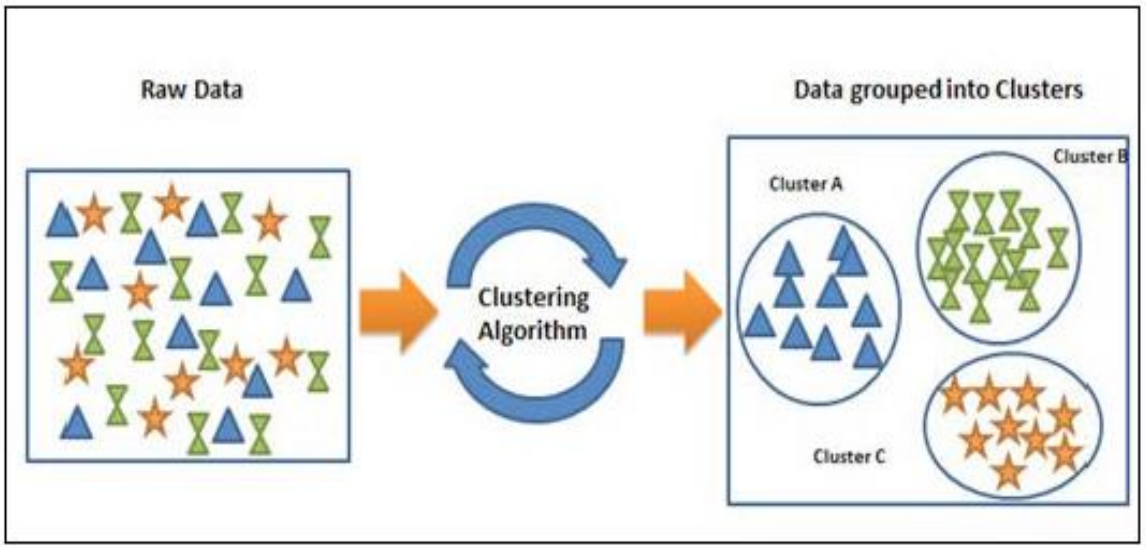


Fuente: Carlos Alba

Para el desarrollo del software se seleccionaron cuatro métodos de clustering, de los cuales dos pertenecen a la categoría por particiones (K-MEANS y CLARA) y los dos restantes hacen parte de clustering jerárquico (AGNES y DIANA).

K-MEANS: Es una técnica de agrupamiento que plantea la división de un conjunto de objetos o datos en k grupos denominados clústeres, en el que cada dato pertenece a un único clúster y se posiciona en él, teniendo en cuenta el valor medio más cercano, los clústeres se forman con base a la distancia mínima (Ver Ilustración 5). En el algoritmo de K-means se determina el número de grupos y centroides, sobre su base el algoritmo identifica los miembros del clúster (dato asignado al grupo) y, a partir de éste recalcula el centroide fundado en los miembros ya identificados. Esta operación es recursiva, por lo tanto se itera hasta que los centroides de cada uno de los clústeres sea único y la distancia entre los datos y el centroide sea óptima, por lo tanto, la precisión de los centroides es la clave para que el agrupamiento basado en particiones pueda tener éxito (Gollapudi, n.d.)

Ilustración 5 - Proceso Clustering



Fuente: Sunila Gollapudi (2016), Practical Machine Learning.

Suponiendo que el conjunto de datos fuese el siguiente:

- D= {x_1 , x_2 ,.. , x_n }, cuando
- Xi= (xi_1,xi_2,…xi_r), es un vector en un espacio de valores reales X ⊆ Rr, y r es el número de atributos en los datos.
- Recodar que el algoritmo k-means divide los datos dados en k clústeres y cada clúster tiene un centro llamado centroide.
- K es especificado por el usuario.

Dado k, el algoritmo k-means funciona de la siguiente manera:

Algoritmo k-means (k, D)

1. Identifique los k puntos de datos como los centroides iniciales (centros de agrupación).
2. Repita el paso 1.
3. Para cada punto de datos x ϵ D hacer.
4. Calcule la distancia de x al centroide.
5. Asigne x al centroide más cercano (un centroide representa un grupo).
6. Fin de la iteración.
7. Vuelva a calcular los centroides utilizando las membresías del grupo actual hasta que se cumpla el criterio de parada.

CLARA: Sus siglas provienen del inglés Clustering Large Applications. El algoritmo CLARA, separa múltiples muestras de la base completa y aplica el algoritmo PAM (Partición Alrededor de Medoids) sobre cada una de ellas; luego, encuentra los conjuntos de k-medoids de las muestras. Teniendo en cuenta que el algoritmo PAM presentaba deficiencia, Kaufman y Rousseuw (1990) decidieron proponer este algoritmo y así poder trabajar con grandes volúmenes de información. Si estas muestras se consideran representativas de toda la base de datos, los medoids de las muestras deberían acercarse a aquellos que se hubiesen escogido de la base de datos completa ("Una revisión de los algoritmos de partición más comunes en el análisis de conglomerados: un estudio comparativo," 2010).

Algoritmo CLARA:

1. Para i = 1 hasta n, repetir los siguientes pasos:
2. Seleccionar una muestra de t objetos de forma aleatoria del conjunto total de datos, y llamar al algoritmo PAM para encontrar k medoids de la muestra.
3. Para cada objeto O_j del conjunto total de datos, determinar cuál de los k medoids es el más similar a O_j.
4. Calcular la disimilitud promedio del agrupamiento obtenido en el paso anterior. Si este valor es menor al mínimo actual, usar este valor como el mínimo actual y retener los k medoids encontrados en el paso (2) como el mejor conjunto de medoids obtenidos.
5. Retornar al paso (1) para comenzar la próxima iteración.

Los métodos de clustering jerárquico se dividen en dos, uno que puede dividir los datos y el otro que puede o unirlos o aglomerarlos, todo depende si la descomposición o agrupación se realiza de arriba hacia abajo o viceversa (Ver Ilustración 6).

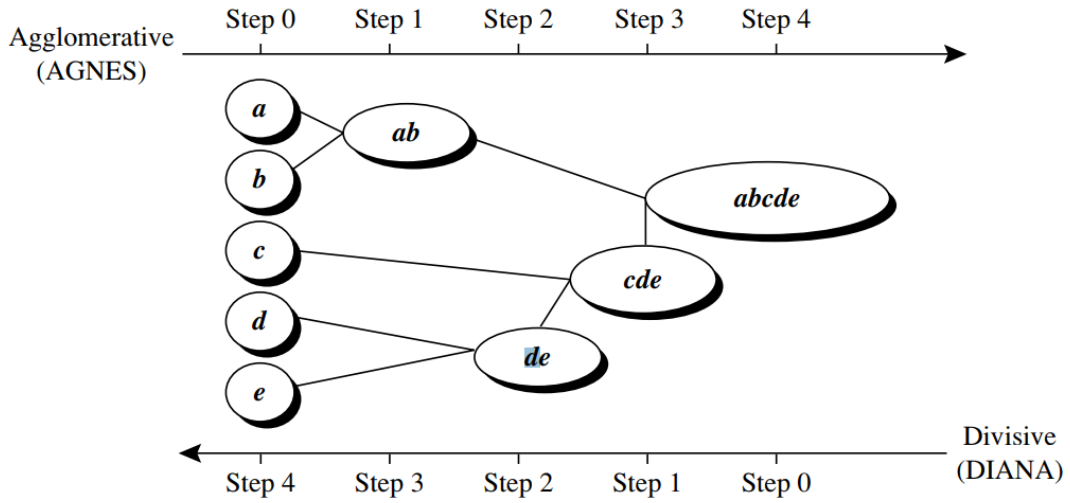
Un método de agrupamiento jerárquico aglomerado utiliza una estrategia en la que generalmente cada objeto o dato, forma su propio clúster y de manera iterativa une grupos y los transforma en grupos cada vez más grandes, hasta que todos los objetos estén en un solo grupo o se cumplan las condiciones ya definidas. El único clúster se convierte en la raíz de la jerarquía (Charu C. Aggarwal & Chandan K. Reddy)

AGNES: (Agglomerative Nesting), es uno de los métodos aglomerantes que consiste en ubicar cada objeto en un grupo propio, los cuales se fusionan paso a paso de acuerdo con algún criterio, cada grupo está representado por todos los objetos en el grupo, y se mide la similitud entre dos grupos por la similitud del par más cercano de puntos de datos que pertenecen a diferentes grupos, el proceso de fusión de clústeres se repite hasta que todos los objetos se fusionan para

formar un solo racimo.

DIANA: (Divisive Analysis) es uno de los métodos de análisis divisivos y su funcionamiento es totalmente inverso a los métodos aglomerantes, en este se crea un grupo inicial con todos los objetos y se procede a dividir el clúster con algún principio, por ejemplo, la distancia euclidiana máxima entre los objetos vecinos más cercanos en el grupo. Este proceso de división se repite hasta que, eventualmente, cada clúster nuevo contiene solo un único objeto, en pocas palabras el método consiste en tener que dividir un clúster grande en varios, teniendo en cuenta parámetros de similitud entre los objetos.

Ilustración 6 - Método Agnes y Diana



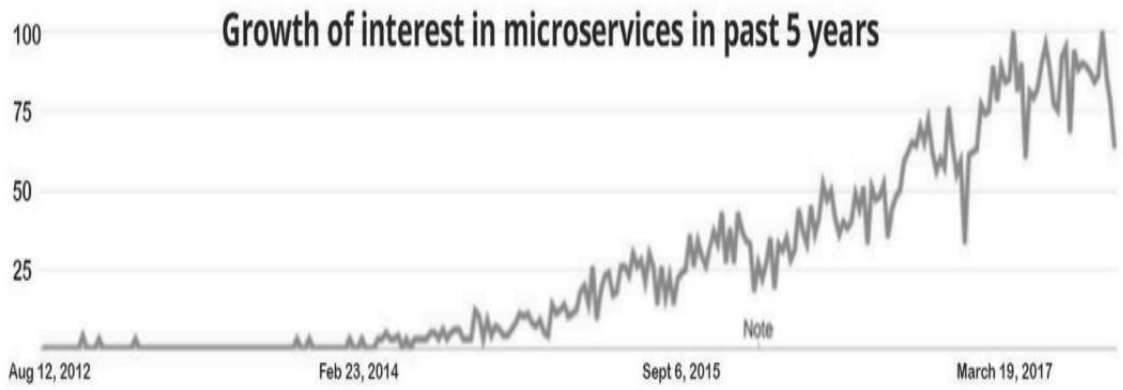
Fuente: DATA MINING Concepts and Tecniques, Jiawey han, Micheline kamber

En la anterior figura (Ver Ilustración 6) se evidencia cómo el método AGNES, inicia asignando un grupo diferente para cada dato y en cada nivel aprovecha la similitud de cada objeto para agruparlos, por otro lado, el método DIANA lleva a cabo un proceso contrario, inicia con un grupo donde contiene todos los datos y los desagrega teniendo en cuenta alguno parámetros.

Microservicios

Para iniciar con este capítulo, es de suma importancia traer en contexto el inicio y evolución del término microservicios, el cual hoy en día se describe como una arquitectura para el desarrollo de software.

Ilustración 7 - Interés en la Arquitectura de Microservicios



Fuente: Google Trends

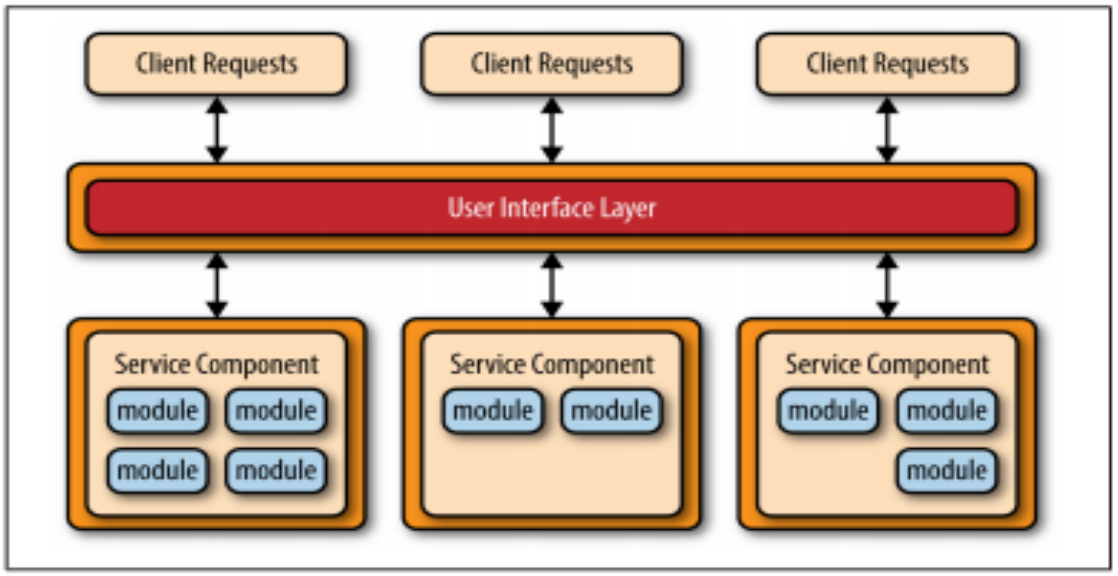
"El interés en la arquitectura de microservicios ha crecido rápidamente desde 2012 como una forma de resolver problemas comunes con aplicación monolítica". *(*Google Trends)

Conforme a lo indicado por Martin Fowler (ingeniero de sistemas británico y orador público en diseño de software empresarial), el término microservicios se inició en un taller de arquitectos de software como una descripción del nuevo campo que los colaboradores estaban explorando. Sin embargo, aún no se evidenciaba una definición en concreto para microservicios, teniendo en cuenta que el término surge como resultado de la experiencia y no sólo como una teoría. En este sentido, una muy buena aproximación lo define como (Zeev Avidan, 2018):

"Pequeños servicios autónomos que trabajan juntos", la estructura de microservicios es una colección de servicios independientes que son estrechos en alcance y se comunican usando protocolos de accesos remotos, los cuales son muy eficientes (Ejemplo: JMS, AMQP, REST, SOAP, RMI, etc.). Reducidos a su esencia, los microservicios son un estilo arquitectónico para desacoplar las funciones comerciales de una arquitectura monolítica tradicional. Cuando también

se encapsula una API en un microservicio, se evitan muchas trampas de diseños monolíticos, es decir, que los servicios son independientes entre sí y cada uno debe desarrollar una funcionalidad de negocio individual (Ver Ilustración 8).

Ilustración 8 - Arquitectura Básica de Microservicios



Fuente: M. Richards, Software Architecture Patterns, O'Reilly Media, Inc, 2015.

Otro concepto fundamental dentro de la arquitectura de microservicios, es que se define como una arquitectura distribuida que logra escalabilidad y tiene características de implementación y desarrollo.

Topologías

Se pueden desarrollar diferentes comportamientos al incluir un patrón de arquitectura de microservicios, pero se recalcan tres topologías principales (Richards, 2015) que son las más comunes:

- Basada en API REST (Representational State Transfer, en español Transferencia de Estado)
- Basada en aplicaciones REST.
- La mensajería centralizada.

Topología basada en API REST: Esta topología es de gran utilidad para sitios web que exponen servicios individuales pequeños y autónomos a través de algún tipo de API, está basada en componentes de servicios de grano fino (por eso se conoce como microservicios), contienen uno o dos módulos que realizan funciones específicas, lo ideal es que cada servicio sea independiente. En esta topología, estos componentes de servicio de grano fino se acceden normalmente mediante una interfaz basada en REST, implementada a través de una capa de API basada en la web desplegada separadamente. Algunos ejemplos de sitios web que exponen servicios de APIS son Google, Amazon, Twitter, Foursquare, etc.

Topología basada en aplicaciones REST: esta topología presenta diferencia con el enfoque basado en API REST puesto que las solicitudes del cliente son recibidas de manera gráfica por medio de pantallas de aplicaciones empresariales tradicionales, basadas en web o en clientes pesados (fat-client) y no sólo por una simple capa de API. La interfaz de usuario de la aplicación se ejecuta como una aplicación web separada, la cual accede de manera remota a componentes de servicios desplegados por separado, por medio de interfaces simples basadas en REST. Otra disimilitud con la topología anterior es que los componentes de servicio son más grandes, de grano más grueso y es una pequeña parte de toda la aplicación. Esta topología es común para las aplicaciones empresariales pequeñas y medianas que tienen un grado de complejidad no muy alto (Richards, 2015) .

La topología de mensajería centralizada: Es equivalente a la topología basada en REST, la diferencia radica en que no utiliza REST para acceso remoto si no un intermediario de mensajes centralizado en los que se encuentra: ActiveMQ, HornetQ, entre otros. Por lo general, en aplicaciones grandes o aplicaciones que requieren un control más sofisticado, sobre la capa de transporte entre la interfaz de usuario y los componentes de servicio se encuentra involucrada esta topología y ofrece una cantidad de ventajas como lo son: mensajería asíncrona, monitoreo, control de errores y mejor balanceo de carga y escalabilidad.

La topología de mensajería centralizada suele confundirse con el patrón SOA o considerarlo "SOA-Lite", pero el agente de mensajes ligeros que se encuentra en esta topología no realiza ninguna transformación, orquestación o enrutamiento complejo, sólo funciona como transporte ligero para acceder a los componentes de servicio remotos.

Ventajas de usar Microservicios

- Facilita la integración de sistemas con tecnologías diferentes.
- Minimiza el tiempo de desarrollo al reducir el impacto en otros componentes del sistema.
- El tiempo de reacción para recuperarse ante un fallo es muy corto.
- Adaptan su escala fácilmente a la demanda del sistema.

Desventajas de usar Microservicios

- Realizar mantenimiento a un sistema distribuido es una tarea con cierto grado de dificultad.
- La consistencia de los datos es asíncrona y no transaccional.
- Probar y monitorear el sistema es algo complejo.

4. MARCO CONCEPTUAL

En el transcurso del proyecto se reconoce múltiples conceptos a grandes escalas, los cuales son fundamentales para el desarrollo de éste, por lo anterior, se hace necesario indicar las percepciones de los ítems más relevantes.

4.1 Big Data

Big Data es el proceso de recolección, análisis y visualización de una amplia variedad de datos, que busca patrones y relaciones entre estos. Big Data hace referencia al mundo de los datos digitales el cual no es posible manejar por técnicas tradicionales. Para obtener un buen resultado es necesario reunir científicos de datos que con algoritmos y bibliotecas de software permitan la distribución, procesamiento y análisis de problemas (Andrew McAfee; Erik Brynjolfsson, 2012). En las principales características del Big Data se encuentra la administración de la gran cantidad de volumen de datos, la variedad en los que se pueden presentar y la velocidad de sus transacciones.

4.2 Científico de Datos

El científico de datos nace gracias al auge del Big Data, se trata de una persona con amplios conocimientos en matemáticas y estadística que además domina la tecnología y, por lo general, diferentes lenguajes de programación. El científico de datos se encarga de hacer descubrimientos al nadar en los datos, son capaces de aportar estructura a grandes cantidades de datos sin forma y hacer posible el análisis (Hernández-Leal et al., 2015). El profesional de la ciencia de datos comunica tendencias obtenidas de los resultados de estudio y ayuda a tomar decisiones.

4.3 Machine Learning

Machine Learning es una rama de la inteligencia artificial que hace uso de la estadística para que los ordenadores tengan la habilidad de aprender automáticamente (Arthur Samuel, 1959), el término ‘aprender’ en el contexto de Machine Learning, se refiere a uno o varios algoritmos capaces de revisar los datos, identificar patrones y predecir comportamientos futuros.

En Machine Learning se identifican tres métodos principales:

- Método de Regresión.
- Método de Clasificación.

- Método de Clusterización (Clustering).

4.4 Análisis De Datos

Es la ciencia que examina datos con la intención de obtener información. El análisis de datos es usado en varias ramas y facilita la toma de mejores decisiones empresariales, también es usado en las ciencias para probar modelos o teorías existentes.

El análisis de datos permite extraer de una medida o de un conjunto de medidas la información que requiere el observador, luego de tener los datos, se continúa con la distribución de los mismos, mediante técnicas determinadas por el analista.

4.5 Clustering

Clustering, en términos de análisis, es una técnica de reducción de datos especializada en interpretar y encontrar subgrupos de observaciones dentro de un conjunto de datos, los cuales se obtienen por la alta similitud de los objetos. El análisis por medio de clustering es usado en diferentes ramas como la política, ciencias, marketing, salud, entre otras. En clustering se identifican dos enfoques de agrupamiento más populares, los cuales son: el agrupamiento jerárquico aglomerado y el particionamiento de agrupamiento. En el primero, cada observación comienza en su propio grupo y luego el algoritmo se encarga de combinar los grupos, dos a la vez, hasta que todos los grupos son fusionados en un sólo clúster. En el clustering por particionamiento de agrupamiento, se especifica el número de grupos buscados (este número usualmente es denominado como K), las observaciones se dividen al azar en grupos K y se reorganizan para formar agrupaciones cohesivas (Kabacoff, 2011).

4.6 Clústeres

Un clúster se define como un conjunto de observaciones, en el cual se asigna un grupo de datos con cierta similitud, que a su vez, son diferentes de los datos de los demás grupos.

4.7 Microservicios

Los microservicios son pequeños servicios autónomos que se comunican entre sí, según Robert C. Martin "se deben reunir esas cosas que cambian por la razón de ser, y separar las cosas que cambian por diferentes razones", este principio es tomado por los microservicios, pues se enfoca el servicio en los límites del negocio, haciendo que sea relevante ejecutar el código para una determinada funcionalidad y al mantener este servicio enfocado en un límite explícito, se evita la tentación de que crezca demasiado, con todas las dificultades asociadas que esto puede causar (Jackson, 2017). En definitiva, una arquitectura de microservicios tiende a desarrollar un software como una figura de pequeños servicios (cada uno se enfoca en una sola actividad) que pueden trabajar en conjunto pero su ejecución es independiente.

5. METODOLOGÍA

Para desarrollar el componente de software se seguirá la metodología de desarrollo XP (Extreme Programming), la cual está fundada en la agilidad y simplicidad, que a diferencia de las metodologías clásicas o tradicionales, busca reducir el número de procesos remplazándolos por el mínimo de procesos cuyo esfuerzo valga la pena, con el objetivo de tener un buen desarrollo de software, más allá de una buena documentación.

En la metodología XP normalmente se diferencian cuatro fases (José Joskowicz, 2008) las cuales son:

- Fase de exploración.
- Fase de planificación.
- Fase de iteraciones.
- Fase de puesta en producción.

Fase de Exploración: Hace uso de las historias de usuario para definir el alcance general del proyecto, esta actividad se realiza junto con los clientes, se evalúan los tiempos iniciales de desarrollo, aunque no sean los finales, teniendo en cuenta que lo que se conoce del proyecto en esta instancia es algo muy general y los requerimientos se pueden modificar en cada iteración. El resultado de esta fase debe ser el tiempo total apreciado para finalizar el software teniendo una visión general del sistema.

Teniendo en cuenta la fase de exploración se realizó el levantamiento de requerimientos funcionales y no funcionales, con lo cual se identificaron temas de clustering para análisis de datos, implementación de microservicios bajo frameworks como ‘spring boot’ y lenguajes de programación como Java y R.

Fase de Planificación: Los requerimientos definidos por el usuario en la fase anterior son utilizados en la fase de planificación en la que se acuerda con los interesados del proyecto el orden en que serán implementados, esta fase es corta a comparación de las demás y el resultado de esta es un plan de entregas.

En la presente fase se identificó que el tiempo máximo de desarrollo del componente es de un semestre académico y que se deben usar herramientas y bases de datos como: Mongo DB, Eclipse, STS Spring Tools y Rserve.

Fase de Iteraciones: Es quizá la fase más importante en el ciclo de la metodología XP. En esta los requerimientos son desarrollados y se hace la entrega, aquí también el usuario interactúa en la presente fase pues se debe

hacer un análisis de la entrega y de lo que se requiere para así retroalimentar y solicitar cambios si es necesario, además en esta fase también es posible medir el avance del proyecto.

En la fase de iteraciones se desarrollan componentes preliminares donde se modifican con cada entrega las API de clustering, API Front, API Web APP, Clustering Script Interface, Clustering Data Base.

Fase de Puesta en Producción: Al finalizar cada iteración se hace entrega de módulos funcionales y sin errores, el cliente puede solicitar no poner el sistema en producción hasta que se obtenga la funcionalidad completa del sistema. En esta fase no se realizan más desarrollos funcionales, pero es necesario estimar tareas de ajuste y soporte.

Para la presente fase ya se conocen los componentes definitivos revisados por los Stakeholders, los cuales se ponen a disposición del usuario en una fase de despliegue.

La metodología XP fue determinada para el desarrollo en parejas (dos personas trabajando en un mismo pc) mejorando notablemente el tiempo de codificación y corrección de errores. Se puede pensar que esta práctica duplicaría el tiempo de desarrollo, pero la verdad es que al trabajar en pares se reducen los errores y se alcanzan mejores diseños (Yolanda, B. L, 2013), compensando la inversión en horas.

Cuando se realiza una codificación en parejas es posible evidenciar grandes ventajas, entre estas:

- Gran parte de los errores se encuentran en el momento en que se codifica, puesto que el código es revisado y evaluado constantemente por dos personas.
- En las pruebas los defectos son menores.
- Los diseños son más eficaces y eficientes con un número menor de líneas de código.
- Cuando finaliza el proyecto existen más personas que conocen al detalle la codificación.
- Si existe un problema al codificar, en pareja se puede resolver más rápido.
- Se aprende del trabajo colectivo y colaborativo haciendo que la información fluya más rápido.

Desventajas del XP:

Es recomendable ser usada en proyectos definidos para un corto plazo.

6. DESARROLLO DEL PROYECTO

Como se indicó anteriormente, este proyecto se basa en el desarrollo de un componente de software. Por lo tanto, a continuación se detallarán las fases para la realización de este proyecto, las cuales se componen de una fase previa de investigación y las fases del desarrollo de software.

6.1 FASE DE INVESTIGACIÓN

Para la realización de este proyecto se realizó previamente una investigación con el fin de comprender el entorno tecnológico y conceptual sobre los últimos avances en tres campos específicos de las ciencias de la computación: Análisis de Datos, Machine Learning (técnicas de clustering) y Microservicios.

El objetivo fundamental de este proyecto es el de desarrollar un componente de clustering que permita realizar análisis de datos, y que pueda ser usado tanto por usuarios finales, como ser integrado a una aplicación global, es decir, ser usado por aplicaciones externas. Por consiguiente, la investigación parte del hecho de analizar la importancia del análisis de datos en la actualidad, ya que es un proceso fundamental y de mucha utilidad, pues el volumen de datos y el acceso a los mismos, es cada vez más creciente y complejo; lo que ha llevado, de manera proporcional, a aumentar la necesidad de extraer la información de estos en todos los campos de estudio.

Por ende, antes de investigar conceptos técnicos, como las tecnologías a usar o las técnicas de clustering, se investigó el estado actual del análisis de datos y de cómo el análisis de clustering toma cada vez más importancia en la generación de información y conocimiento en cualquier campo, ya que desde el ‘data mining’ hasta el Big Data ha estado presente. Conocer estos conceptos de análisis de datos es fundamental al momento de comprender el análisis de clustering, lo cual permite que el desarrollo del software sea más ágil.

Posteriormente, se realizó la investigación sobre las principales técnicas de clustering, abordando temas como conceptos, propósitos, funcionalidades, y usos, tanto a nivel investigativo como en el mercado e implementaciones, con el fin de seleccionar las técnicas más utilizadas y cuya complejidad de implementación fuese mínima, con el fin de cumplir con el tiempo estipulado para el proyecto.

Es así como se determinó escoger las cuatro técnicas de clustering y el lenguaje de desarrollo, el cual es R; ya que en la revisión académica, conceptual y bibliográfica realizada, éste es el lenguaje al que los autores más hacen referencia, y que a consideración del tiempo de desarrollo es realmente óptimo, pues ofrece paquetes ya preestablecidos en los cuales se encuentran las funciones necesarias para desarrollar las técnicas de clustering.

Adicionalmente, y con el fin de seleccionar la arquitectura para el diseño del componente, se optó por investigar sobre una de las arquitecturas con mayor crecimiento y aceptación en los últimos años, los microservicios. Estos permiten la implantación de aplicaciones en componentes más pequeños, lo cual, a su vez, garantiza la escalabilidad, ya que pueden ser implementados fácilmente en diferentes ambientes, desde contenedores, hasta soluciones cloud o 'host standalone'. Lo anterior, permite que el componente a desarrollar tenga la versatilidad de ser implementado bajo cualquier ambiente y que pueda operar de manera independiente, es decir, sin estar contenido dentro de una aplicación global.

Finalmente, para el desarrollo del microservicio, se hará uso de tecnologías ampliamente conocidas como el lenguaje de programación Java, ya que de acuerdo al último ranking realizado por TIOBE, sigue siendo el lenguaje de programación más popular con un 15.77%; lo cual garantiza que el componente de software que despliega el microservicio, pueda ser ajustado mediante mantenimiento evolutivo, sin preocuparse por el conocimiento acerca del lenguaje de programación. Adicionalmente, Java y su característica portable, garantizan su implementación bajo cualquier ambiente, pues al contener el framework 'Spring Boot', permite realizar la construcción de microservicios, al embeber dentro del framework un servidor web.

6.2 FASE DE ANÁLISIS Y PLANIFICACIÓN

Para el análisis y planificación, primero se partió de la investigación para indagar y definir los conceptos claves que permitieran abordar el desarrollo de los componentes. Posteriormente, basados en el análisis de requerimientos, se procedió a establecer cuáles eran aquellas funcionalidades, restricciones y necesidades del componente, dando como resultado una base sólida, la cual se estructuró en el documento anexo 'Documento de Especificación de Requerimientos de Software (SRS)', en el cual se establecieron las funcionalidades con las que debe cumplir el componente, éstas son las siguientes:

- Debe basarse en la arquitectura de microservicios.
- Ejecutar algoritmos de clustering a demanda del usuario o aplicación.
- Parametrizar cada algoritmo de clustering.
- Exponer una API para el consumo y parametrización del componente.
- Exponer una interfaz web para el cargue y configuración de datos por parte del usuario.

Adicionalmente, se establecieron las consideraciones y limitantes, como los lenguajes de programación a usar, frameworks y tecnologías. Para más información acerca del análisis de requerimientos, consultar el anexo (SRS).

En cuanto a la planificación, se planteó que el proyecto tuviera una duración máxima de cuatro meses y cuyos entregables fueran iterativos, es decir, que cumplieran con la metodología XP. En este sentido, se priorizó la codificación, para que dependiendo de la complejidad, se dividieran los requerimientos en partes más pequeñas y manejables, de tal manera que un sólo requerimiento pudiera ser abordado en más de una etapa iterativa.

En definitiva, los requerimientos funcionales (RF) y no funcionales (RNF) determinados son:

Funcionales:

- Cargue de datos vía web.
- Ejecución de algoritmos de clustering.
- Funcionalidad de parametrización de algoritmos de clustering (interfaz web).
- Funcionalidad de parametrización de algoritmos de clustering (API).
- Validar la ejecución del algoritmo (WEB).
- Validar la ejecución del algoritmo (API).

No funcionales:

- Interfaz web intuitiva.
- Disponibilidad independiente del sistema general.
- Ejecución asíncrona.
- Seguridad dependiente de la aplicación global.

- Manuales de implementación.
- Manual de usuario.
- Lenguaje de programación de la API.
- Framework de programación de la API.
- Lenguaje de algoritmos de clustering.
- Interfaz entre Java y R.
- Lenguaje interfaz web.
- Base de datos para persistencia.
- Algoritmos base de clustering.

6.3 FASE DE DISEÑO

Con el 'Documento de Especificación de Requerimientos de Software (SRS)' ya establecido se realizó el diseño de software partiendo desde dos conceptos clave:

Descripción del diseño: Esta parte fue basada en el estándar IEEE 1016-2009 (IEEE, 2009) 'Documento de Diseño de Software' (SDD), en el cual se contemplaron aspectos del diseño del componente, basados en los requerimientos establecidos en el SRS, con el fin de tener una visión clara sobre el componente a desarrollar. En dicho documento se pudieron establecer las entidades claves en el diseño del componente, como las tecnologías a usar, las dependencias, patrones de diseño (inyección de dependencias, inversión de control y patrón strategy), para más información ver Anexo 2 (SDD). Adicionalmente, se determinaron los siguientes componentes de diseño:

- Clustering API.
- Clustering Front API.
- Clustering Web App.
- Clustering Script Interface.
- Clustering Scripts.
- Clustering Data Base.

Para más información acerca de los descriptores de diseño de software, consultar el anexo 'Documento de Diseño de Software' (SDD).

Arquitectura de Software: Una vez establecidos los descriptores de diseño, se procedió a realizar la arquitectura de software, la cual se describió en el 'Documento de Arquitectura de Software' (SAD) basado en el estándar IEEE 1471

(Requirements, 2012). En este documento se plantearon las diferentes vistas arquitectónicas que describen el comportamiento del componente, para lo cual se utilizó la metodología de ***Modelo de "4+1" Vistas de la Arquitectura de Software,*** la cual permite detallar los diferentes puntos de vista para la descripción adecuada y estructural del comportamiento del software. Del documento SAD se pueden extraer los siguientes diagramas que describen el comportamiento (Software & Kruchten, 2006), para más información consultar el anexo SAD.

Para más información acerca de la arquitectura de software, consultar el anexo 'Documento de Arquitectura de Software' (SAD).

6.4 FASE DE DESARROLLO

Para esta fase se tuvieron en cuenta los documentos estipulados en la fase de diseño y se dividió esta fase en dos etapas: Preparación y Codificación.

Preparación

En esta etapa se preparó el entorno necesario para la programación o codificación del componente de software. Para el ambiente fueron necesarias las siguientes herramientas:

- **Repositorio de datos**: Con el fin de garantizar un control de versiones y de centralizar el almacenamiento del código fuente, se utilizó el repositorio 'github', allí se crearon los proyectos que se describen en la siguiente tabla:

Tabla 1 - Repositorio de datos

Proyecto	Tipo	Descripción
clustering_microservice_api	Proyecto Java	Contiene el código fuente del Clustering API
clustering_microservice_front_api	Proyecto Java	Contiene el código fuente del Front API
clustering_service	Proyecto R	Contiene los scripts de Clustering
clustering_web_content	Proyecto Angular	Contiene el código fuente de la Interfaz Web

- **Herramienta de gestión de dependencias**: Para los proyectos Java se requiere gestionar las dependencias, por lo tanto se usó **Maven** como gestor de dependencias.

- **IDE**: Para el proyecto Java se usó el IDE **STS** (basado en eclipse) de Spring, que es especial para los proyectos de Spring Boot. Para el proyecto R se usó el IDE **RStudio** y para el proyecto Angular se usó el IDE **Visual Studio Code**.

- **Gestor de Base de Datos**: Se usó el gestor Studio 3T para administrar la base de datos Mongo DB.

Codificación

Durante esta etapa se realizó el desarrollo del componente, generando así cuatro componentes principales, Clustering API, Clustering Front API, Clustering Web App y Clustering Scripts. Adicionalmente, se desarrolló el modelo de datos cuya definición de modelo se encuentra en el documento SDD. Sin embargo, es importante explicar que para el modelo de datos se optó por un modelo 'No Relacional' (No-SQL), debido a que no es necesario, en primera medida, tener datos estructurados y dichos datos no necesariamente contienen los mismos parámetros que puedan normalizarse.

A continuación se describe el desarrollo de cada componente:

Clustering API

Este es el microservicio principal, el cual se encarga de manejar la lógica de clustering y se desarrolló en Java bajo el framework **Spring Boot**. La responsabilidad de este componente es la de exponer una API REST que permita ejecutar los scripts donde se encuentran los algoritmos de clustering (desarrollados en R), para eso la API cuenta con una serie de servicios y utilitarios que permiten desde el código desarrollado en Java, enviar parámetros, recibir respuestas y ejecutar código en R, lo anterior, a través de un servidor TCP/IP (**RServe**) el cual expone una interfaz que puede ser utilizada mediante las librerías **Rengine** y **RServe** (las cuales mediante **Maven** se integran dentro del proyecto).

A continuación y en la siguiente imagen se muestran los paquetes que componen el Clustering API, en los cuales se encuentran las clases que contienen el código

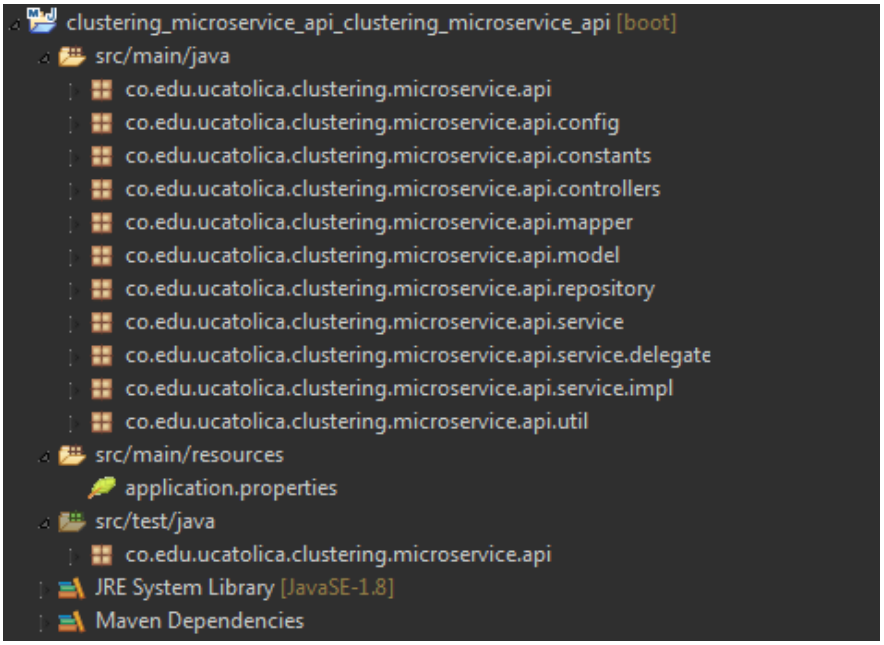
responsable de su funcionamiento, éstas se agruparon por responsabilidades, conteniendo cada paquete lo siguiente:

Tabla 2 - Descripción Paquetes Clustering API

Paquete	Descripción
co.edu.ucatolica.clustering.microservice.api	Contiene la clase principal de Spring Boot que se encarga de inicializar la aplicación.
co.edu.ucatolica.clustering.microservice.api.config	Contiene las clases que configuran desde el inicio los servicios de Mongo DB y los Thread pool para la ejecución asíncrona.
co.edu.ucatolica.clustering.microservice.api.controllers	Contiene dos controladores (uno para ejecutar métodos y otro para realizar operaciones) con sus respectivos endpoints encargados de resolver las peticiones, estos endpoints son los siguientes:
	clustering-api/run/kmeans: el cual permite ejecutar el algoritmo kmeans
	clustering-api/run/pam: el cual permite ejecutar el algoritmo pam
	clustering-api/run/clara -> el cual permite ejecutar el algoritmo clara
	clustering-api/hierarchical/{agnes|diana}: el cual permite ejecutar el algoritmo jerárquico y en cuyo path se estable el tipo ya sea agnes o diana
	clustering-api/operations/method/all permite extraer todos los métodos disponibles y sus parámetros
	clustering-api/operations/find/execution/{Id} permite consultar una ejecución previa mediante el Id.
co.edu.ucatolica.clustering.microservice.api.mapper	Contiene las clases encargadas de mapear entre modelos.

co.edu.ucatolica.clustering.microservice.api.model	Contiene los modelos (POJO) encargados de contener los datos.
co.edu.ucatolica.clustering.microservice.api.repository	Contiene los Repositorios de Spring Data para realizar las operaciones CRUD en la base de datos.
co.edu.ucatolica.clustering.microservice.api.service	Contiene las interfaces que definen los servicios.
co.edu.ucatolica.clustering.microservice.api.service.delegate	Es contiene la clase que es el servicio principal que reúne los otros servicios y utilitarios para actuar como fachada.
co.edu.ucatolica.clustering.microservice.api.service.impl	Contiene los servicios implementados
co.edu.ucatolica.clustering.microservice.api.util	Contiene los utilitarios que son utilizados como por ejemplo el proveedor o fábrica de clases concretas de la Interfaz **IClusteringAlgorithm** que encapsula la lógica de los algoritmos de clustering y que se basa en el patrón **Strategy**.

Ilustración 9 - Paquetes Clustering API



Fuente: Joseph Rubio

Clustering Front API

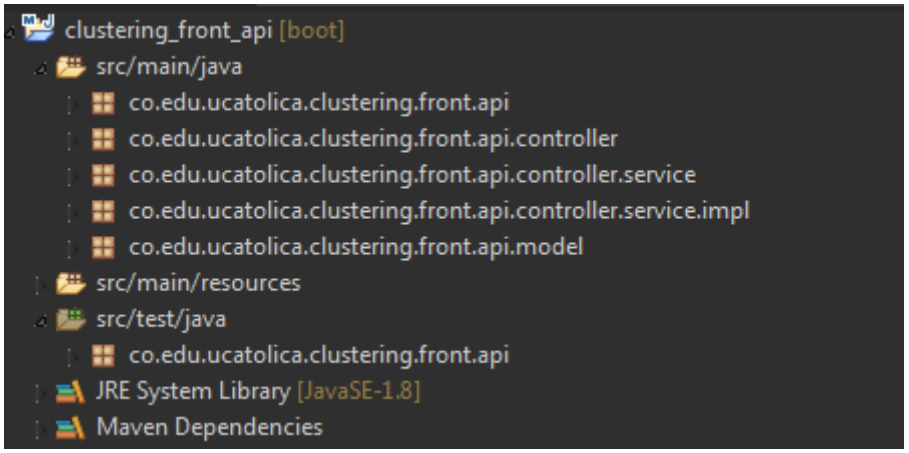
Este es el microservicio que actúa como backend para la Clustering Web App, fue desarrollado en Java y bajo el framework Spring Boot. La responsabilidad de este microservicio es la de consumir el microservicio Clustering API y dotar a la aplicación web (Clustering Web App) de las funcionalidades necesarias para gestionar la interacción con el usuario, tales como el cargue de datos, la selección de métodos de clustering y sus respectivos parámetros, y la consulta de ejecuciones de clustering previas. A continuación se describen los paquetes:

Tabla 3 - Descripción Paquetes Front API

Paquete	Descripción
co.edu.ucatolica.clustering.front.api	Contiene la clase principal de Spring Boot que se encarga de inicializar la aplicación.
co.edu.ucatolica.clustering.front.api.controller	Este paquete contiene los controladores cuyos endpoints son los que proveen la funcionalidad a la aplicación web. A continuación se detallan los endpoints:
	clustering-front-api/check/clustering-exec/{Id} : permite consultar por Id si la ejecución de un análisis de clustering ya terminó, de ser así retornará un link para descargar los resultados, en caso contrario retornará un mensaje informativo
	clustering-front-api/data/upload: permite cargar el archivo CSV con los datos a analizar.
	clustering-front-api/download/{Id}: permite descargar los resultados del análisis.
	clustering-front-api/methods/all: permite consultar los métodos y sus configuraciones.

co.edu.ucatolica.clustering.front.api.controller.service	Contiene la definición de los servicios principales como el **IClusteringClientService** encargado de comunicarse mediante la funcionalidad de **Spring ResTemplate** con el Clustering API.
co.edu.ucatolica.clustering.front.api.controller.service.impl	Contiene la implementación de los servicios.
co.edu.ucatolica.clustering.front.api.model	Contiene los modelos (POJO) encargados de contener los datos.

Ilustración 10 - Paquetes Clustering Front API



Fuente: Joseph Rubio

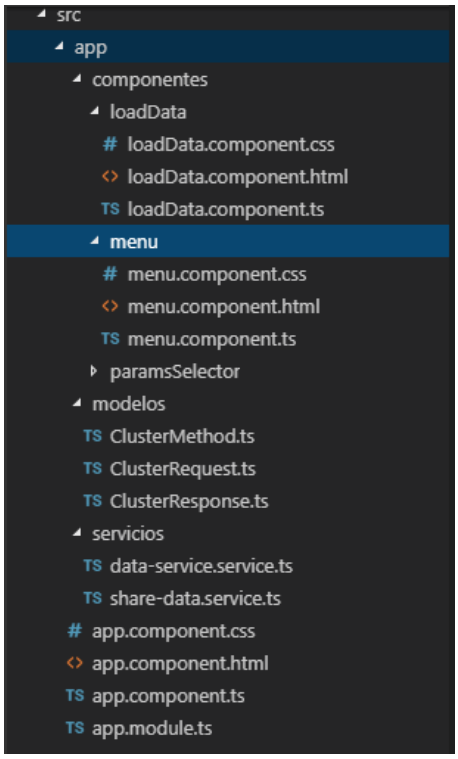
Clustering Web App

Esta es la estructura de la aplicación web, la cual fue desarrollada bajo el framework Angular 5 (TypeScript, JavaScript, HTML5 y CSS3) y representa al frontend y sus responsabilidades son las de exponer la interfaz gráfica para la interacción por parte del usuario y al ser desarrollada en angular es completamente modular lo cual facilitará futura mejoras. A continuación se describe el contenido y estructura del Clustering Web App:

Tabla 4 - Descripción Estructura Aplicación Web

Carpeta	Descripción
App	Esta es la carpeta raíz de la aplicación y contiene el módulo principal que integra los componentes.
Componentes	Esta carpeta contiene los componentes que conforman la aplicación web, los cuales en este caso son:
	Menú: encargado de crear el menú y sus acciones.
	loadData: encargado de crear el componente para cargue de datos.
	paramsSelector: encargado de crear el componente que permite seleccionar los parámetros por cada tipo de algoritmo.
	En esta carpeta se encuentran los modelos que contienen los datos.
Modelos	**ClsuteringMethod**: este modelo es el que contiene la información del método de lcutering específico y sus parámetros.
	ClusteringRequest: Este modelo contiene los datos para realizar la petición al backend
	ClusteringResponse: Este modelo contiene los datos que representan la respuesta del backend.
Servicios	**data-service:** este es el servicio encargado de gestionar las peticiones http al backend (Clustering Front API)
	share-data: Este es el servicio que permite compartir datos entre los componentes de Angular.

Ilustración 11 - Estructura Clustering Web App



Fuente: Joseph Rubio

Clustering Scripts

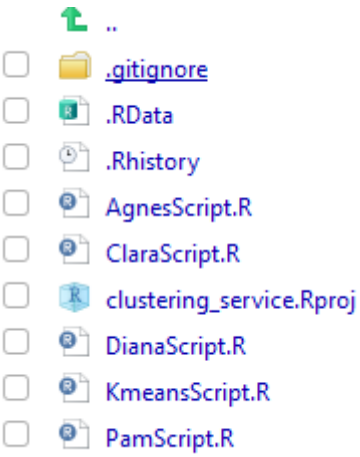
Son los scripts que contiene los algoritmos de clustering, sus responsabilidades son la de recibir la información (desde RServe, información enviada por el Clustering API) así como por cuestiones de diseño, serializar la respuesta de clustering en JSON, lo cual permite un manejo mucho más adecuado y sencillo de esta respuesta desde el clustering API. A continuación se describen los scripts.

Tabla 5 - Descripción Clustering Scripts

Script	Descripción
KmeansScript	Contiene la implementación del algoritmo K-Means, proveído por la librería **amap**.
PamScript	Contiene la implementación del algoritmo PAM, proveído por la librería **cluster**.

ClaraScript	Contiene la implementación del algoritmo CLARA, proveído por la librería **cluster**.
AgnesScript	Contiene la implementación del algoritmo jerárquico **AGNES**, proveído por la librería **cluster**.
DianaScript	Contiene la implementación del algoritmo jerárquico **DIANA**, proveído por la librería **cluster**.

Ilustración 12 - Clustering Scripts



Fuente: Joseph Rubio

6.5 FASE DE PRUEBAS

Para realizar las pruebas en el componente, se utilizó el enfoque de 'pruebas de caja blanca', también llamadas pruebas estructurales que tienen como fin realizar las pruebas únicamente de los componentes que son propios de la funcionalidad ISTQB (ISTQB, 2018).

El nivel de pruebas seleccionado fue el de 'pruebas unitarias' por las características del componente, cuya funcionalidad se encuentra centralizada en el Clustering API y por los tiempos del desarrollo, se decidió que las pruebas unitarias otorgaran el nivel necesario de confianza y calidad para probar el componente de software.

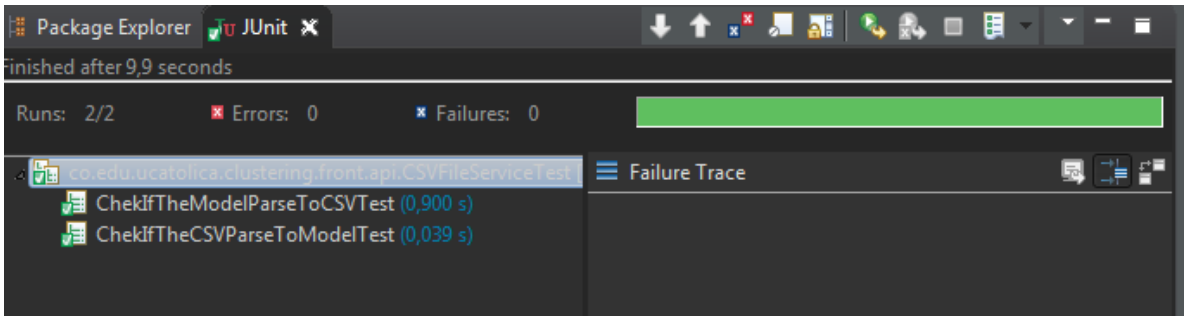
De acuerdo con lo anterior, se establecieron los siguientes casos de prueba:

Tabla 6 - Casos de Prueba

Caso de Prueba	Descripción	Resultado	Componente
chekIfTheCSVParseToModelTest	Esta prueba permite validar mediante un archivo CSV simulado (String separado por comas), si la conversión del CSV al modelo correspondiente es correcta.	Exitoso	Clustering Front API
chekIfTheModelParseToCSVTest	Esta prueba permite validar mediante un archivo CSV simulado (String separado por comas), si la conversión del modelo un archivo CSV es correcta.	Exitoso	Clustering Front API
execKMeansTest	Esta prueba permite probar desde la integración entre Java y R hasta el Script del algoritmo **Kmeans**, los datos utilizados son el paquete de datos de prueba **"USArrests"** que R ofrece.	Exitoso	Clustering API
execPAMTest	Esta prueba permite probar desde la integración entre Java y R hasta el Script del algoritmo **PAM**, los datos utilizados son el paquete de datos de prueba **"USArrests"** que R ofrece.	Exitoso	Clustering API

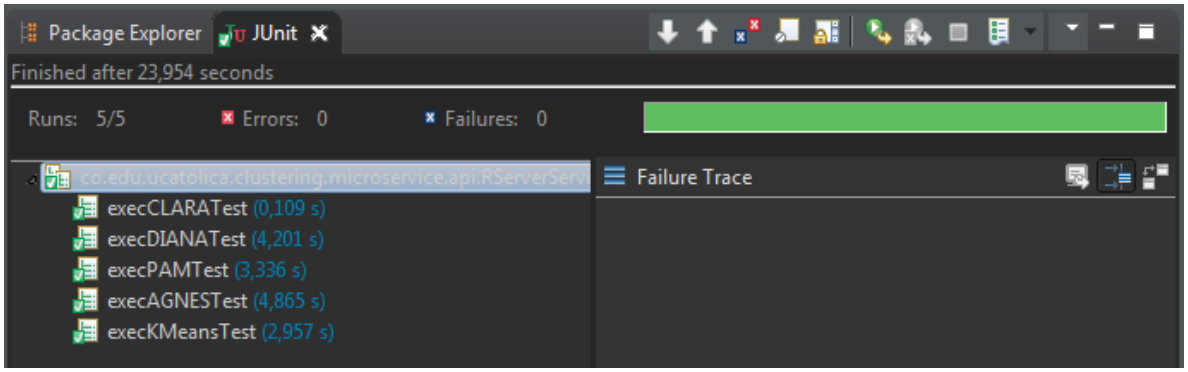
execCLARATest	Esta prueba permite probar desde la integración entre Java y R hasta el Script del algoritmo **CLARA**, los datos utilizados son el paquete de datos de prueba **"USArrests"** que R ofrece.	Exitoso	Clustering API
execAGNESTest	Esta prueba permite probar desde la integración entre Java y R hasta el Script del algoritmo jerárquico **AGNES**, los datos utilizados son el paquete de datos de prueba **"USArrests"** que R ofrece.	Exitoso	Clustering API
execDIANATest	Esta prueba permite probar desde la integración entre Java y R hasta el Script del algoritmo jerárquico **DIANA**, los datos utilizados son el paquete de datos de prueba **"USArrests"** que R ofrece.	Exitoso	Clustering API

Ilustración 13 - Unit Test Clustering Front API



Fuente: Joseph Rubio

Ilustración 14 - Unit Test Clustering API



Fuente: Joseph Rubio

6.6 FASE DE DOCUMENTACION

Para esta fase y con el fin de no incurrir en documentación exhaustiva (dando cumplimiento a lo estipulado en la metodología XP) y que no reste tiempo al desarrollo, lo que se referencia a continuación son las directrices que a nivel de documentación se tuvieron:

- Se hará a nivel de código.
- Debe incluir descripción de propiedades, métodos y clases.
- Debe incluir un encabezado que indique el propietario (en este caso la Universidad Católica de Colombia) y la fecha de desarrollo.
- Debe contener el nombre y correo del autor.

- Debe indicarse la versión.
- Debe ser en español.
- Los nombres de las propiedades, clases, enumeraciones, métodos e interfaces, sin importar el idioma, debe ser lo más semántico posible.

Esta documentación se llevó a cabo durante el proceso de desarrollo.

6.7 FASE DE IMPLEMENTACION

En esta fase no se hace mucho énfasis ya que el componente únicamente fue desplegado en ambiente local, y el entregable es el código fuente. Sin embargo, para la implementación del componente se deben tener las siguientes consideraciones:

- Host y puertos para los microservicios **Clustering API** y **Clustering Front API**.
- Host y puertos para el servidor **Rserve.**
- Host y puertos para la base de datos **Mongo DB**.
- Repositorio para los **Clustering Scripts**.

Lo anterior, para que puedan ser implementados, en cuyo caso deben cambiarse dentro del código fuente, en el archivo de propiedades (**application.properties**).

6.8 FASE DE MANTENIMIENTO

Si bien dentro del proyecto esta fase no aplica, ya que es un entregable, la estructura, documentación y desarrollo del código fuente, permite que de ser necesario un mantenimiento correctivo o evolutivo, pueda ser realizado sin ningún contratiempo, ya que se usaron los principios básicos de diseño de software, cumpliendo con GRASP y SOLID.

7. RESULTADOS

Una vez culminado el desarrollo del componente se obtuvieron los resultados correspondientes que se generaron mediante los casos de prueba (Ver **5.5. FASE DE PRUEBAS**), dichos resultados persistieron dentro de la base de datos (Mongo DB), puntualmente en la colección ***execution,*** de acuerdo con lo descrito en el **SDD** (Ver Anexo 2), en donde se define que esta colección almacena los resultados de las ejecuciones de los algoritmos de clustering.

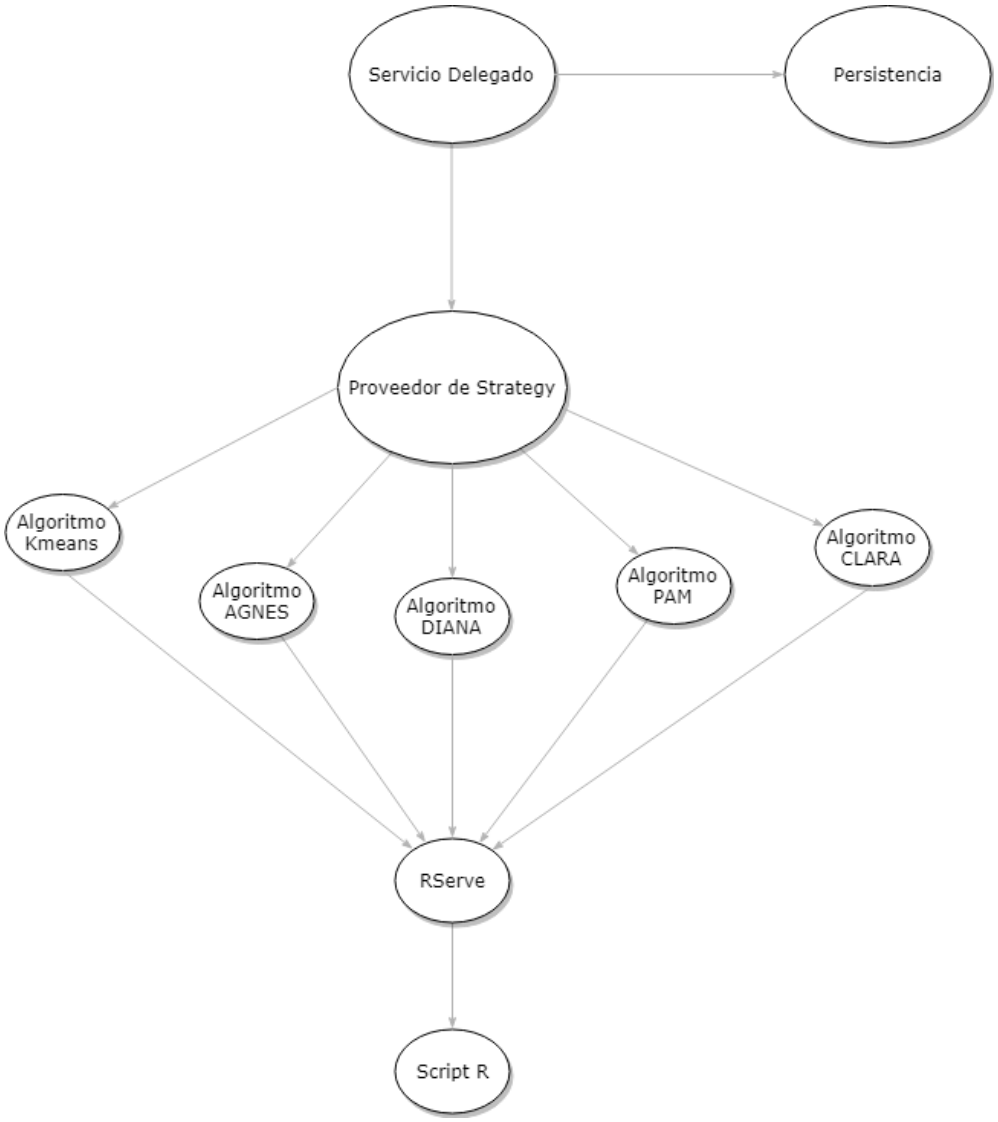
Cabe resaltar que en esta sección se hace mención de los resultados de los casos de prueba para el **Clustering API**, ya que los casos de prueba del **Clustering Front API** se realizaron mediante **Mocks** (con la ayuda del framework de pruebas **Mockito**), el cual permite simular objetos con sus respectivas entradas y salidas, con el fin comprender el comportamiento y el flujo de un componente. En el caso del **Clustering Front API**, se simuló el mapeo entre un archivo **CSV** (simulado por un **String** de caracteres) y el modelo correspondiente, y de igual manera se simuló el mapeo contrario, es decir, de un modelo a un archivo **CSV**. Estas pruebas también se apoyaron en **Junit** para determinar la validez de los resultados esperados, en este caso los resultados arrojados por los mocks, como se evidencian en la sección **5.5. FASE DE PRUEBAS,** tuvieron un resultado exitoso.

Sin embargo, para el caso del **Clustering API,** las pruebas unitarias no se realizaron a través de mocks, más bien se realizaron utilizando datos de prueba y que como se menciona en los casos de prueba, se utilizó el data set "**USArrests**". Estas pruebas se enfocaron en determinar que el algoritmo de clustering se ejecutara y devolviese el valor correspondiente, el cual sería guardado en la colección, es decir el alcance se centró en los resultados.

Estas pruebas comprendieron un flujo en el cual desde un **servicio delegado** se llama al **proveedor de Strategy** que contiene los diferentes objetos (construidos mediante el patrón ***Strategy***), y que representan la lógica de los **algoritmos de clustering,** estos a su vez consultan a **Rserve** para ejecutar el **Script R** correspondiente, y finalmente, el flujo de datos vuelve hasta el delegado mediante retornos hasta que el servicio **persiste** el resultado en la base de datos.

En la siguiente ilustración (Ver Ilustración 13) se puede evidenciar el flujo de pruebas.

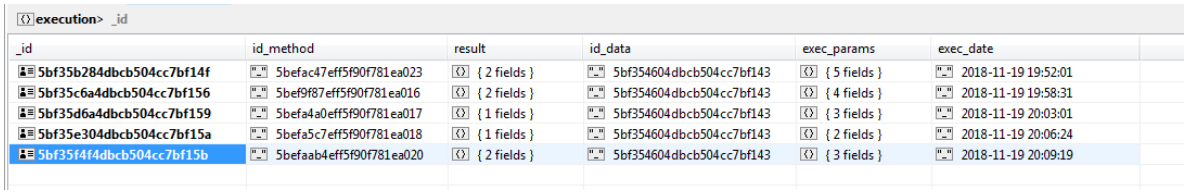
Ilustración 15 - Flujo de Pruebas Clustering API



Fuente: Joseph Rubio

Finalmente, al consultar la colección se evidencian los datos persistidos (Ver Ilustración 14 – Colección Execution). También se puede apreciar la estructura de datos JSON del resultado de cada una de las pruebas (Ver Ilustración 15 – Resultado Ejecución Algoritmo Clustering).

Ilustración 16 - Colección Execution



execution> _id

_id	id_method	result	id_data	exec_params	exec_date
5bf35b284dbcb504cc7bf14f	5befac47eff5f90f781ea023	{ 2 fields }	5bf354604dbcb504cc7bf143	{ 5 fields }	2018-11-19 19:52:01
5bf35c6a4dbcb504cc7bf156	5bef9f87eff5f90f781ea016	{ 2 fields }	5bf354604dbcb504cc7bf143	{ 4 fields }	2018-11-19 19:58:31
5bf35d6a4dbcb504cc7bf159	5befa4a0eff5f90f781ea017	{ 1 fields }	5bf354604dbcb504cc7bf143	{ 3 fields }	2018-11-19 20:03:01
5bf35e304dbcb504cc7bf15a	5befa5c7eff5f90f781ea018	{ 1 fields }	5bf354604dbcb504cc7bf143	{ 2 fields }	2018-11-19 20:06:24
5bf35f4f4dbcb504cc7bf15b	5befaab4eff5f90f781ea020	{ 2 fields }	5bf354604dbcb504cc7bf143	{ 3 fields }	2018-11-19 20:09:19

Fuente: Joseph Rubio

Ilustración 17 - Resultado Ejecución Algoritmo Clustering

Document JSON Viewer


```
            "Louisiana" : 6.0,
            "Maine" : 6.0,
            "Maryland" : 6.0,
            "Massachusetts" : 6.0,
            "Michigan" : 6.0,
            "Minnesota" : 6.0,
            "Mississippi" : 6.0,
            "Missouri" : 6.0,
            "Montana" : 6.0,
            "Nebraska" : 6.0,
            "Nevada" : 6.0,
            "New Hampshire" : 6.0,
            "New Jersey" : 6.0,
            "New Mexico" : 6.0,
            "New York" : 6.0,
            "North Carolina" : 6.0,
            "North Dakota" : 6.0,
            "Ohio" : 6.0,
            "Oklahoma" : 6.0,
            "Oregon" : 6.0,
            "Pennsylvania" : 6.0,
            "Rhode Island" : 6.0,
            "South Carolina" : 6.0,
            "South Dakota" : 6.0,
            "Tennessee" : 6.0,
            "Texas" : 6.0,
            "Utah" : 6.0,
            "Vermont" : 6.0,
            "Virginia" : 6.0,
            "Washington" : 6.0,
            "West Virginia" : 6.0,
            "Wisconsin" : 6.0,
            "Wyoming" : 6.0
        }
    },
    "id_data" : "5bf354604dbcb504cc7bf143",
    "exec_params" : {
        "clusters" : 6.0,
        "clustering_method" : "average",
        "distance_method" : "binary"
    },
    "exec_date" : "2018-11-19 20:03:01",
    "_id" : ObjectId("5bf35d6a4dbcb504cc7bf159")
```


Fuente: Joseph Rubio

8. CONCLUSIONES

Sobre el desarrollo de este proyecto para el Componente Web Parametrizable de Clustering para Análisis de Datos, se puede concluir lo siguiente:

Actualmente, en el mercado no existe un componente de software que permita realizar clustering integrado a otras aplicaciones o que pueda ser ejecutado por sí solo. Así mismo, las técnicas para análisis de datos son herramientas muy efectivas al momento de extraer información relevante que permita, basado en datos, comprender y/o agrupar las características de los mismos.

El clustering es una de las técnicas más idóneas para encontrar nueva información dentro de un conjunto de datos, que permita generar un nuevo conocimiento. Para realizar el clustering se debe tener debidamente preparados y estructurados los datos analizar, es importante tener en cuenta que no se pueden analizar datos no estructurados, ejemplo: videos, audios, imágenes, etc.

La mayoría de los algoritmos de clustering pueden ser organizados dentro de una interfaz común, ya que el conjunto de datos de entrada y sus parámetros son similares, lo que facilita el desarrollo de un componente genérico de software basado en clustering. Desacoplar la lógica de la aplicación y la lógica del clustering permite una alta reusabilidad y escalabilidad, ya que son ambas son independientes.

Usar el lenguaje de programación R es de gran ayuda en el proceso de desarrollo del software y en especial para el clustering, ya que la mayoría de las funcionalidades que se requieren para la implementación ya se encuentran creadas. En el mismo sentido, utilizar la arquitectura de micro servicios permite que el componente pueda ser usado a gran escala en aplicaciones más robustas.

Finalmente, usar la metodología de desarrollo XP facilita en gran medida el cumplimiento de los compromisos estipulados en los requerimientos de software.

9. BIBLIOGRAFÍA

Andrea, A. D. ', Ferri, F., & Grifoni, P. (2015). *Approaches, Tools and Applications for Sentiment Analysis Implementation*. *International Journal of Computer Applications* (Vol. 125). Retrieved from http://messenger.yahoo.com/features/emoticons

Andrew McAfee; Erik Brynjolfsson. (2012). *Big Data The Management Revolution*. Retrieved from https://pdfs.semanticscholar.org/02c7/740af5540f23a2da23d1769e64a8042ec62e.pdf

Cathy O'Neil, R. S. (2013). *Doing Data Science_ Straight Talk from the Frontline-O'Reilly Media (2013)*.

Doug Laney. (2001). *Application Delivery Strategies*. Retrieved from https://blogs.gartner.com/doug-laney/files/2012/01/ad949-3D-Data-Management-Controlling-Data-Volume-Velocity-and-Variety.pdf

Everitt, B. S., Landau, S., Leese, M., & Stahl, D. (n.d.). *Cluster Analysis 5th addition2011*. https://doi.org/10.1007/BF00154794

Garner, S. R. (n.d.). *WEKA: The Waikato Environment for Knowledge Analysis*. Retrieved from https://www.cs.waikato.ac.nz/~ml/publications/1995/Garner95-WEKA.pdf

Gollapudi, S. (n.d.). *PRACTICAL MACHINE LEARNING*.

Hall, M., Frank, E., Holmes, G., Pfahringer, B., Reutemann, P., & Witten, I. H. (n.d.). *The WEKA Data Mining Software: An Update*. Retrieved from https://www.kdd.org/exploration_files/p2V11n1.pdf

Hernández-Leal, E. J., Duque-Méndez, N. D., Moreno-Cadavid, J., Hernández-Leal, E. J., Duque-Méndez, N. D., Moreno-Cadavid, J., & Big, ". (2015). *Big Data: una exploración de investigaciones, tecnologías y casos de aplicación Big Data: an exploration of research, technologies and application cases* (Vol. 20). Retrieved from http://www.redalyc.org/pdf/3442/344251476001.pdf

IEEE. (2009). *IEEE Std 1016-2009 (Revision of IEEE Std 1016-1998), IEEE Standard for Information Technology—Systems Design—Software Design Descriptions*. *Middle East* (Vol. 2009). https://doi.org/10.1109/IEEESTD.2009.5167255

ISTQB. (2018). International Software Testing Qualifications Board ISTQB ®

Certified Tester: Foundation Level Extension Syllabus Agile Tester, 1–77.

Jackson, N. (2017). *Building Microservices with Go*. O'Reilly Media.

Jagla, B., Wiswedel, B., & Coppée, J.-Y. (2011). Extending KNIME for next-generation sequencing data analysis. *BIOINFORMATICS APPLICATIONS NOTE*, *27*, 2907–2909. https://doi.org/10.1093/bioinformatics/btr478

José Joskowicz. (2008). *Reglas y Prácticas en eXtreme Programming*. Retrieved from https://iie.fing.edu.uy/~josej/docs/XP - Jose Joskowicz.pdf

Kabacoff, R. I. (2011). *R IN ACTION: Data analysis and graphics with R*. *Livro*. https://doi.org/citeulike-article-id:10054678

KAUSHIK, S. (2016). An Introduction to Clustering & different methods of clustering. Retrieved September 30, 2018, from https://www.analyticsvidhya.com/blog/2016/11/an-introduction-to-clustering-and-different-methods-of-clustering/

Lin, N. P., Chang, C. I., Chueh, H. E., Chen, H. J., & Hao, W. H. (2008). A deflected grid-based algorithm for clustering analysis. *WSEAS Transactions on Computers*, *7*(3), 125–132.

Mitchell, T. M. (1997). *Machine Learning*. (Reviews, Ed.).

Moujahid, A. (n.d.). Clustering, 1–17.

RapidMiner | Sistemas de Minería de Datos | Software de Minería de Datos. (n.d.). Retrieved August 31, 2018, from https://www.microsystem.cl/plataforma/rapidminer/

Requirements, Q. (2012). *International Standard Iso / Iec* (Vol. 25021). https://doi.org/10.1109/IEEESTD.2015.7106438

Richards, M. (2015). *Software Architecture Patterns*. (O. Media, Ed.).

Software, A., & Kruchten, P. (2006). Planos Arquitectonicos : El Modelo de " 4 + 1 " Vistas de la La Arquitectura L ´, *12*(6), 1–16.

Timmerman, B. (2012). Flow measurements in patient-specific conducting airways models. *16th Int Symp on Applications of Laser Techniques to Fluid Mechanics*, *2*(3), 9–12. https://doi.org/10.1109/TETC.2014.2330519

Una revisión de los algoritmos de partición más comunes en el análisis de conglomerados: un estudio comparativo. (2010).

10. ANEXOS

ANEXO 1 (SRS)

DOCUMENTO DE ESPECIFICACIÓN DE REQUERIMIENTOS DE SOFTWARE (SRS)

COMPONENTE WEB PARAMETRIZABLE DE

CLUSTERING PARA ANÁLISIS DE DATOS (CWPCAD)

JOSEPH ALEXANDER RUBIO TAPIAS

CARLOS ANDRÉS ALBA RODRÍGUEZ

ESPECIFICACIÓN DE REQUERIMIENTOS

DIEGO ALBERTO RINCÓN YÁÑEZ MCSc

TABLA DE CONTENIDO

Control de Cambios

Fecha	Autor	Versión	Comentarios
21/10/2018	Joseph Rubio	1.0	Documento SRS
19/11/2018	Joseph Rubio	2.0	Ajuste final

11. INTRODUCCIÓN

Resumen

El presente es un Documento de Especificación de Requerimientos de Software (SRS por sus siglas en inglés) en el que se detallan los requerimientos, las características y las funcionalidades del Componente Web Parametrizable de Clustering para Análisis (CWPCAD), con el fin de establecer y limitar el alcance del mismo y su funcionalidad para conocimiento del usuario.

Propósito

Delimitar la información necesaria para el diseño y desarrollo del (CWPCAD), indicando sus alcances, funcionalidades y restricciones. Este documento va dirigido a arquitectos, desarrolladores y coordinadores de proyectos de software.

Autores

Nombre	Joseph Alexander Rubio Tapias
Rol	Software Developer
Categoría Profesional	Ingeniero
Contacto	*315-8712775*

Nombre	Carlos Andrés Alba Rodríguez
Rol	Software Developer
Categoría Profesional	Ingeniero
Contacto	*311-8903630*

Nombre	Diego Alberto Rincón Yáñez
Rol	Arquitecto de Software

Categoría Profesional	Ingeniero
Contacto	*300-5707966*

Ámbito del Sistema

Actualmente, la Universidad Católica de Colombia, tiene la necesidad de realizar análisis de datos para estudios de investigación, apoyándose en técnicas de Machine Learning. Sin embargo, no cuentan con un software debidamente estructurado y unificado para esta labor, razón por la cual se desarrollará un componente de software basado en clustering (técnica de Machine Learning) que pueda integrarse a una aplicación para estos estudios.

Este componente inicialmente lo que va a permitir es usar cuatro técnicas de clustering y éstas podrán ser parametrizadas por un usuario final. Adicionalmente, estará disponible para que de manera práctica se puedan agregar nuevas técnicas de clustering en el futuro. Finalmente, expondrá una API para integraciones con otros sistemas.

Definiciones, Acrónimos y Abreviaturas

Definiciones

Clustering	Es una técnica de reconocimiento de patrones que se caracteriza por identificar grupos en un conjunto de datos, sin que se le indique a priori las variables para dicho agrupamiento. Es conocido como aprendizaje no supervisado.
Micro servicio	Es un tipo de arquitectura en donde las funcionalidades de un componente o aplicación se desagregan en funcionalidades más pequeñas e independientes.
API	Application Program Interface. Es un conjunto de métodos expuestos para ser consumidos por una aplicación.
Algoritmo	Es un conjunto de pasos ordenados que permite realizar operaciones o solucionar problemas específicos.

Acrónimos

DERS	Documento de Especificación de Requerimientos de Software
CWPC	Componente Web Parametrizable de Clustering para Análisis
RF	Requerimiento Funcional
RNF	Requerimiento No Funcional

Abreviaturas

Visión General del Documento

Alcance

Diseño y desarrollo del (CWPCAD) Componente Web Parametrizable de Clustering para Análisis de Datos.

Este será un componente que expondrá sus funcionalidades a través de tecnologías web y permitirá ejecutar algoritmos de clustering, basado en una entrada de datos (previamente definida por el usuario) y que retornará un resultado.

Este componente permitirá lo siguiente:

- Debe basarse en la arquitectura de micro servicios.
- Exponer un API con los métodos de clustering.
- Disponer de una interfaz web para el cargue de datos por parte del usuario.
- Parametrizar cada algoritmo de clustering.

Descripción

Funciones del producto

En términos generales, el componente debe permitir lo siguiente:

- **Debe basarse en la arquitectura de micro servicios:** El diseño debe estar basado en la arquitectura de micro servicios con el fin de garantizar un desacople de cualquier aplicación monolítica y ser independiente.

- **Ejecutar algoritmos de clustering a demanda del usuario o aplicación:** El componente debe recibir una orden para ejecutar un algoritmo de clustering con unos parámetros y datos entregados por el usuario o aplicación. El usuario los debe enviar vía web, y las aplicaciones mediante el API.

- **Añadir nuevos algoritmos de clustering:** Como se indicó anteriormente, sólo se dispone de cuatro algoritmos inicialmente, pero la aplicación debe permitir agregar nuevos algoritmos para realizar la ejecución.

- **Parametrizar cada algoritmo de clustering:** Cada algoritmo tiene un conjunto de parámetros propios de su funcionamiento, los cuales pueden ser personalizados dependiendo del tipo de ejecución, por lo cual es necesario crear dichos parámetros desde la web o desde la API.

- **Exponer una API para el consumo y parametrización del componente:** Para realizar las funcionalidades de ejecución, parametrización y creación de algoritmos de clustering, es necesario contar con una API que pueda ser consumida por aplicaciones externas en donde se puedan enviar los datos de manera estandarizada y así mismo, obtener una respuesta positiva o un conjunto de errores por parte de la API.

- **Exponer una interfaz web para el cargue y configuración de datos por parte de un usuario:** La interfaz web debe permitir a un usuario final cargar un archivo con la información necesaria para la ejecución del algoritmo de clustering y también contar con las funciones necesarias para crear nuevos algoritmos y parametrizarlos.

Características de los Usuarios

Los usuarios que van a utilizar este componente deben tener las siguientes características:

- Conocimientos sobre las características necesarias que debe tener el conjunto de datos a cargar.
- Debe ser capaz de familiarizarse con el componente de manera intuitiva.
- Debe conocer el tipo de resultado que arrojará el componente.
- Debe poder parametrizar los algoritmos de clustering.

Restricciones

- Se debe ejecutar en ambientes que tengan instalada una versión de java igual o superior a 1.8.
- Los datos que se pasarán al componente para ejecutar los algoritmos de clustering deben estar debidamente estandarizados, es decir, de carácter numérico o categórico.
- El componente es una API interna, por ende, no debe ser expuesta en Internet.
- Dependiendo de las características del hardware (procesador, memoria) donde el componente se ejecutará, así mismo será la velocidad de respuesta de la aplicación, por ende no se garantizan tiempos a priori del componente.

Suposiciones y Dependencias

- Se asume que el entorno de ejecución del componente debe cumplir con las características que se mencionarán en los requerimientos no funcionales. Ejemplo: versión de java, versión del servidor R, etc.
- Se asume que se cuenta con un equipo de hardware donde el componente va a estar corriendo, además de contar con los puertos habilitados y los permisos necesarios para su ejecución.
- Se asume que el componente es una API privada, es decir, no se expondrá en Internet, por lo tanto, no contará con seguridad de acceso.

Requisitos futuros

- El componente, al ser basado en arquitectura de micro servicios, estará disponible para ser instalado o ejecutado en contenedores tipo Docker.
- Debido a la tecnología en la que se desarrolla, si se desea exponer al público, se podrá agregar seguridad sin mayor inconveniente.

Metodología

Para realizar el componente se plantea el uso de una metodología ágil de desarrollo de software, la cual se conoce como XP (Extreme Programming), con el fin de garantizar el entregable dentro del tiempo estimado, además de permitir la entrega iterativa del desarrollo del componente. Adicionalmente, esta metodología plantea el desarrollo por parejas con el objetivo de poder detectar errores y garantizar una mayor calidad en el código.

Requerimientos Específicos
Funcionalidad

A continuación, se detallan los requisitos que el sistema deberá implementar.

Código	**RF-1**
Nombre de requerimiento	Cargue de Datos vía Web.
Prioridad	
Descripción	El componente debe permitir el cargue de datos mediante un archivo Excel, a través de una interfaz web.
Entradas	Archivo CSV
Proceso	Cargue de datos en el componente
Salidas	Mensaje de ejecución fallida o en proceso.

Código	**RF-2**
Nombre de requerimiento	Ejecución de algoritmos de clustering.
Prioridad	
Descripción	El componente debe ejecutar algoritmos de clustering dependiendo de la selección del usuario o aplicación externa (API).
Entradas	Archivo CSV, estructura de datos tipo JSON.
Proceso	Ejecución dentro del componente del algoritmo seleccionado con los datos de entrada.
Salidas	Mensaje de ejecución fallida o en proceso (en el caso de ser una aplicación web), estructura de datos tipo JSON (en el caso de un API).

Código	**RF-3**
Nombre de requerimiento	Funcionalidad de parametrización de algoritmos de clustering (interfaz web).
Prioridad	
Descripción	Se debe disponer de una interfaz web para que el usuario pueda ingresar los parámetros de los algoritmos de clustering.
Entradas	Formulario web
Proceso	Persistir los parámetros dentro de la base de datos
Salidas	Mensaje de ejecución fallida o completada.

Código	**RF-4**
Nombre de requerimiento	Funcionalidad de parametrización de algoritmos de clustering (API)
Prioridad	
Descripción	Se debe disponer de una API para que pueda ser consumida por una aplicación externa y así pueda ingresar los parámetros de los algoritmos de clustering.
Entradas	JSON
Proceso	Persistir los parámetros dentro de la base de datos.
Salidas	Mensaje de ejecución fallida o completada.

Código	**RF-5**
Nombre de requerimiento	Validar la Ejecución del algoritmo (WEB).
Prioridad	
Descripción	Se debe disponer de una funcionalidad que permita consultar si un proceso de clustering fue ejecutado.
Entradas	Formulario Web
Proceso	Consultar en a Base de datos si el proceso ya culmino.
Salidas	Archivo CSV con los resultados, mensaje de ejecución fallida o incompleta.

Código	**RF-6**
Nombre de requerimiento	Validar la Ejecución del algoritmo (API).
Prioridad	
Descripción	Se debe disponer de una funcionalidad que permita consultar si un proceso de clustering fue ejecutado.
Entradas	JSON
Proceso	Consultar en a Base de datos si el proceso ya culmino.
Salidas	JSON con los resultados, mensaje de ejecución fallida o incompleta.

Usabilidad

Código	**RNF-1**
Nombre de requerimiento	Interfaz web intuitiva.
Prioridad	
Descripción	Se debe disponer de una interfaz web que tenga un tiempo de entendimiento no mayor a 2 horas.

Disponibilidad

Código	**RNF-2**
Nombre de requerimiento	Disponibilidad independiente del sistema general.
Prioridad	
Descripción	El componente siempre debe estar disponible independientemente a que la aplicación global esté en funcionamiento (microservicio externo).

Desempeño

Código	**RNF-3**
Nombre de requerimiento	Ejecución asíncrona.
Prioridad	
Descripción	El componente debe realizar la ejecución de sus funcionalidades de manera asíncrona para evitar indisponibilidad por time out.

Seguridad

Código	**RNF-4**
Nombre de requerimiento	Seguridad dependiente de la aplicación global.
Prioridad	
Descripción	El componente no debe tener ningún tipo de seguridad a nivel de autenticación y autorización, estos deben ser gestionados desde la aplicación global.

Soporte

Código	**RNF-5**
Nombre de requerimiento	Manuales de implementación.
Prioridad	
Descripción	Se debe entregar un manual donde se expliquen los pasos y requisitos necesarios para implementar el componente.

Código	**RNF-6**
Nombre de requerimiento	Manual de usuario y tipo de errores.
Prioridad	
Descripción	Se debe entregar un manual donde se detalle la parametrización y uso del componente (en la interfaz web) y los posibles mensajes de error que pueda generar.

Negocio

Código	**RNF-7**
Nombre de requerimiento	Lenguaje de programación de la API
Prioridad	
Descripción	Se utilizará el lenguaje de programación java para realizar la programación de la API (versión mayor a 1.8)

Código	**RNF-8**
Nombre de requerimiento	Framework de programación de la API
Prioridad	
Descripción	Se utilizará el framework spring boot en versión mayor a 2.0

Código	**RNF-9**
Nombre de requerimiento	Lenguaje de algoritmos de clustering.
Prioridad	
Descripción	Se utilizará el lenguaje de programación R para realizar los algoritmos de clustering.

Código	**RNF-10**
Nombre de requerimiento	Interfaz entre java y R.
Prioridad	
Descripción	Para comunicar estos dos lenguajes se debe utilizar el servidor RServe que permite la ejecución de los

	algoritmos de clustering desde java.

Código	**RNF-11**
Nombre de requerimiento	Lenguaje interfaz web.
Prioridad	
Descripción	Se debe utilizar el framework angular 5 para realizar la programación de la interfaz web. Este framework utiliza el lenguaje typescript.

Código	**RNF-12**
Nombre de requerimiento	Base de datos para persistencia.
Prioridad	
Descripción	Se debe utilizar una base de datos no relacional, para este caso se selecciona Mongo DB.

Código	**RNF-13**
Nombre de requerimiento	Algoritmos base de clustering.
Prioridad	
Descripción	El componente por defecto incluirá cuatro algoritmos de clustering listos para usar y los cuales son, esto con el fin de aprovechar las librerías pre-construidas de R: • Kmeans • Hierarchical (Agnes y Diana) • Pam (Kmedoids) • Clara

Interfaces

El componente dispondrá de dos tipos de interfaces:

- **Interfaz Web:** Es el punto de entrada en formato HTML, para que el usuario pueda realizar el cargue, parametrización, ejecución y recepción de resultados.

- **API:** Son los métodos expuestos para realizar el cargue, parametrización, ejecución y recepción de resultados mediante la arquitectura REST, por parte de aplicaciones externas.

Requerimientos de Licenciamiento

No aplica porque se hará uso de tecnologías 'open source' en el diseño y desarrollo del componente.

Glosario

Clustering	Es una técnica de reconocimiento de patrones que se caracteriza por identificar grupos en un conjunto de datos, sin que se le indique a priori las variables para dicho agrupamiento. Es conocido como aprendizaje no supervisado.
Micro servicio	Es un tipo de arquitectura en donde las funcionalidades de un componente o aplicación se desagregan en funcionalidades más pequeñas e independientes.
Machine Learning	Es una rama de la computación y de la Inteligencia Artificial que se encarga de desarrollar técnicas que permitan a las máquinas aprender y ejecutar operaciones autónomas.
API	Application Program Interface. Es un conjunto de métodos expuestos para ser consumidos por una aplicación.

Algoritmo	Es un conjunto de pasos ordenados que permite realizar operaciones o solucionar problemas específicos.
CRUD	Create Read Update and Delete. Es una funcionalidad a nivel de base de datos que permite crear, leer, actualizar y borrar.
Interfaz	Es el medio que permite una fácil interacción del usuario con el software usualmente está compuesta por menús, ventanas y elementos intuitivos para un uso cómodo.
Monolítico	Sistemas que cuentan con un solo núcleo, conformado por estructuras fijas que trabajan entre sí. Son sistemas que están escritos en un solo código lógico.
Parametrizar	Hace referencia a la configuración de un sistema teniendo en cuenta unos datos específicos
Lenguaje R	Es un lenguaje de programación y ambiente de desarrollo enfocado al análisis estadístico y gráfico, funciona en los sistemas operativos como GNU/LINUX, Windows, Macintosh y UNIX.
Contenedor	Es un objeto que contiene otros objetos los cuales pueden removerse o ser agregados, la aplicación funciona como un paquete en el que se puede incluir bibliotecas y dependencias.
Docker	Es un proyecto de código abierto que permite generar contenedores de software en los que se automatiza el despliegue de software.
Extreme Programming	Es una metodología ágil de desarrollo que se preocupa más por la adaptabilidad o corrección de funcionalidad que por la previsibilidad, permite cambios de requisitos sobre la marcha.
Iterativo	Es la repetición de un proceso específico dentro de un sistema
JSON	JavaScript Object Notation, es un formato de texto ligero para intercambiar datos. Y es una alternativa a XML
Persistir	Es la operación que permite salvaguardar la información de

	un objeto de forma permanente y poder leerla para usarla en el momento que se requiera.
Asíncrono	Es cuando un evento no tiene lugar en total correspondencia temporal con otro evento.
Time Out	Medida que indica a un programa el tiempo máximo que debe esperar para concluir una función o tarea específica.
Framework	Conjunto de conceptos y criterios el cual conforma la estructura de un sistema que puede ser útil en el desarrollo de software.
Java	Es un lenguaje de programación orientado a objetos que permite ejecutar un código y este no tiene que ser recompilado para correr en otra plataforma. También hace referencia a una plataforma de software para computadores.
Spring Boot	Es una herramienta que simplifica el desarrollo de aplicaciones basadas en el framework spring, su finalidad es que el desarrollador solo se concentre en el core del negocio y no se preocupe por la configuración de spring.
RServe	Es un software comercial que se puede ejecutar en un conjunto de plataformas, a mayor escala y carga de trabajo, y en topologías cliente/servidor que soportan acceso remoto sobre conexiones seguras.
Angular 5	Es una nueva versión del framework Angular JavaScript para aplicaciones web, el cual sirve para construir aplicaciones siguiendo el patrón MVC que se ejecutan en el navegador.
TypeScript	Es un superconjunto de JavaScript que se compila a un JavaScript simple, de programación libre y de código abierto, es usado para desarrollar aplicaciones JavaScript que se ejecutarán en el lado del cliente o del servidor (Node.js).
Mongo DB	Es una base de datos orientada a documentos y no es relacional (NoSql), por lo cual no es necesario seguir un esquema, los documentos son almacenados en BSON, que es una representación binaria de JSON.

Kmeans	Es una técnica de agrupamiento que plantea la división de un conjunto de objetos o datos en k grupos denominados clústeres, en el que cada dato pertenece a un único clúster y se posiciona en él, teniendo en cuenta el valor medio más cercano, los clústeres se forman con base a la distancia mínima.
Hierarchical (agnes y diana)	Es un método de agrupación jerárquica que tiene dos tipos aglomerante y divisivo. • **Aglomerante:** Cada objeto inicia siendo su propio grupo y cada vez que sube de nivel se fusiona con los demás verificando sus características. • **Divisivo:** Es lo contrario al aglomerante, todos los objetos inician haciendo parte de un grupo y cuando bajan de nivel se van separando dependiendo de sus características.
Pam (kmedoids)	(Partición Alrededor de Medoids) Es un algoritmo de clustering que divide el conjunto de datos en grupos, intentando minimizar la distancia entre los puntos etiquetados en un grupo y un punto designado como el centro de ese grupo.
Clara	Clustering Large Applications. El algoritmo CLARA, separa múltiples muestras de la base completa y aplica el algoritmo PAM (Partición Alrededor de Medoids) sobre cada una de ellas; luego, encuentra los conjuntos de k-medoids de las muestras.
HTML	HyperText Markup Language (lenguaje de marcas de hipertexto), es un lenguaje de marcado el cual se usa para el desarrollo de páginas web siguiendo el estándar del World Wide Web.
REST	(Representational State Transfer), no es una arquitectura de software, es un conjunto de restricciones con las que se facilitan crear un estilo de arquitectura de software para la comunicación entre cliente y servidor.
Open Source	Es un modelo de desarrollo de software donde el código

	fuente hace parte del dominio público (por lo que puede ser modificado y publicado) y este no tiene ningún valor comercial.

ANEXO 2 (SDD)

DOCUMENTO DE DISEÑO DE SOFTWARE (SDD)

COMPONENTE WEB PARAMETRIZABLE DE CLUSTERING PARA ANÁLISIS DE DATOS (CPWCAD)

JOSEPH ALEXANDER RUBIO TAPIAS

CARLOS ANDRÉS ALBA RODRÍGUEZ

DOCUMENTO DE DISEÑO DE SOFTWARE

DIEGO ALBERTO RINCÓN YÁÑEZ MCSc

TABLA DE CONTENIDO

Control de Cambios

Fecha	Autor	Versión	Comentarios
28-10-2018	Joseph Rubio	1.0	Documento SDD
19/11/2018	Joseph Rubio	2.0	Ajuste final

1. Introducción

En el presente Documento de Diseño de Software (SDD por sus siglas en inglés), (Plantilla estándar IEEE 1016 – 2009) se define la Documentación del desarrollo del Componente Web Parametrizable de Clustering para Análisis de Datos (CWPCAD) en el que se detalla la funcionalidad y relación entre los diferentes artefactos de software que lo componen. Lo anterior, basado en el Documento de Especificación de Requerimientos de Software (SRS).

1.1. Alcance

Delimitar y estructurar el diseño de los artefactos de software para el CWPCAD, el cual es un componte de software que permitirá realizar análisis de datos mediante algoritmos de clustering, de acuerdo con lo contemplado en el SRS.

Entre sus características principales se encuentran:

- Acceder mediante una interface web (para un usuario final)
- Acceder mediante un API (aplicación externa)
- Cargue y envió de datos a los cuales se les realizara el análisis mediante la selección de un método de clustering (previamente seleccionado los parámetros necesarios para el método seleccionado)
- Ejecución del método de clustering seleccionado.

1.2. Propósito

El propósito de este documento es el de describir la estructura del diseño del CPWCAD bajo los lineamientos establecidos en el SRS para lo cual se abordarán los siguientes temas que permitirán estructura el diseño:

- Elementos de Diseño
- Restricciones y Limitaciones
- Diseño de flujo de datos
- Diseño del modelo de datos
- Diseño de interface de usuario

1.3. Público objetivo

El presente documento va dirigido a:

Desarrolladores: con el fin de poder otorgar una mayor comprensión del diseño y características del CPWCAD durante la etapa de desarrollo del mismo.

Arquitectos de Software: con el fin de permitir definir arquitecturas que sustente el diseño estipulado en este documento.

1.4. Referencias

Carlos, M. C. J., & Rojas, O. (n.d.). Patrones de Diseño Patrones de Diseño.

El patrón de inyección de dependencia y su utilidad - Arquitectura Java. (n.d.). Retrieved October 30, 2018, from https://www.arquitecturajava.com/el-patron-de-inyeccion-de-dependencia/

Hernández-piña, L. N., & Jaimez-gonzález, C. R. (1870). Serialización de objetos PHP a XML Serialization of PHP Objects to XML, 125(2016), 87–95.

Inversión de control, por @davidvalverde. (n.d.). Retrieved October 30, 2018, from http://www.davidvalverde.com/blog/inversion-de-control/

Proebsting, T. A., Townsend, G., Bridges, P., Hartman, J. H., Newsham, T., & Watterson, S. A. (n.d.). Toba: Java for Applications A Way Ahead of Time (WAT) Compiler. Retrieved from

https://www.usenix.org/legacy/publications/library/proceedings/coots97/full_papers/proebsting/proebsting.pdf

2. Definiciones

Patrones de Diseño	Son técnicas que permiten solucionar errores o problemas típicos y recurrentes en el desarrollo de software. Los patrones permiten codificar conocimientos, estilos y principios existentes y que se han probado que es válido.

Dataframe	La función dataframe crea marcos de datos, colecciones de variables estrechamente acopladas que comparten muchas de las propiedades de las matrices y las listas, utilizadas por la mayoría de los programas de modelado de R como la estructura de datos fundamental.
Serialización	La serialización es el proceso de escribir un objeto en un medio de almacenamiento como un archivo, con el fin de enviarlo a través de una red.
Standalone	Es un software independiente que no requiere otro paquete de software para ser ejecutado, tampoco requiere de internet u otro proceso informático.
Inyección de Dependencias.	Es un patrón de diseño orientado a objetos, en donde se proporcionan objetos a una clase la cual no crea dichos objetos. Esos objetos cumplen contratos que necesitan nuestras clases para poder funcionar.
Inversión de Control	La inversión de control es una forma de programar en la que el flujo de ejecución de un programa se invierte con relación a los métodos de programación tradicionales, en los que la interacción se expresa de forma imperativa realizando llamadas a procedimientos o funciones.

3. Descripción del contenido de diseño

A continuación, se detalla en los siguientes ítems que permiten describen el diseño del software.

3.1. Stakeholders de diseño

Son aquellos stakeholders que se encargaran del diseño del CWPCAD, este rol lo cumplen los desarrolladores de software los cuales tendrán a cargo entre otras responsabilidades las siguientes tareas:

- Desarrollar los componentes del software garantizando su reusabilidad y buenas prácticas de codificación.
- Desarrollar la interfaz de usuario.
- Desarrollar el modelo datos y su acceso.
- Dar mantenimiento correctivo y evolutivo al CWPAD.

3.2. Vistas de diseño

Representan los diagramas que describen el software, cada vista permite entender el diseño de las características principales del software en este caso el CWPCAD. Estos diagramas serán diseñados mediante UML.

3.3. Puntos de vista de diseño

Son las diferentes vistas que describen una característica o punto de vista del software.

Las siguientes son las vistas seleccionadas para describir el diseño del software:

- **Punto de vista de composición**: este representa el diseño e interacción de los principales componentes del software.
- **Punto de vista de información**: este representa el diseño del modelo de datos.
- **Punto de vista de Interfaz**: este representa el diseño de la interface de usuario.

3.4. Elementos de diseño

Son las entidades, atributos, relaciones y restricciones que hacen parte del diseño del CWPCAD.

3.4.1 Entidades de diseño

A continuación se detalla las entidades de diseño, las cuales están fundamentadas en el SRS.

Entidades	Descripción
Patrones de Diseño de Software	Se utilizan los siguientes patrones: ✓ Inyección de dependencias. ✓ Inversión de control. ✓ Patrón Strategy.

Lenguaje de Programación API	El lenguaje de programación para las API será Java.
Lenguaje de Programación Clustering	El lenguaje de programación para los scripts de clustering será R.
Framework API	El framework para la desarrollo de las API será Spring Boot.
Framework Aplicación Web	El framework para el desarrollo de la aplicación web será Angular.
Sistema de Base de Datos	El sistema de base de datos a utilizar será Mongo DB.
Dependencias de software	Dentro de la construcción del software se requerirá de las siguientes dependencias: **Backend** ✓ Versión de Java 1.8 o superior. ✓ Versión de R igual o superior a 3.4.2. ✓ Versión RServe igual o superior 1.7.3 ✓ Framework Spring Boot igual o superior a 2.0. ✓ Apache commos igual o superior a 1.6 ✓ Versión Junit 4 o superior. **FrontEnd** ✓ Versión de Angular 5 o superior. ✓ Versión de Bootstrap 4 o superior. **Base de datos** ✓ Versión Mongo DB 3.6 o superior.
Clustering API	Este componente es el encargado de exponer la API para que diferentes aplicaciones puedan ejecutar

	algoritmos de clustering. Se basa en una API REST creada bajo el framework Spring Boot. Las siguientes características representan un endpoint: ✓ **Listar métodos de clustering:** Esta responsabilidad es la que se encarga de devolver todos los métodos de clustering (con sus parámetros). ✓ **Ejecutar métodos de clustering:** Esta responsabilidad es la que se encarga de recibir los datos de entrada para la ejecución asíncrona del algoritmo de clustering seleccionado. ✓ **Validar el procesamiento de los algoritmos de clustering:** Esta responsabilidad es la que se encarga de validar que el procesamiento asíncrono haya terminado. ✓ **Retornar los resultados:** Esta responsabilidad es la que se encarga de devolver los resultados una vez ejecutado el algoritmo de clustering seleccionado.
Clustering Front API	Este componente es la API que expondrá los métodos que utilizará la aplicación web. Las siguientes características representan un endpoint: ✓ **Permitir cargar archivos CSV:** El archivo

	contendrá los datos que el usuario cargará para ejecutar el algoritmo de clustering. ✓ **Listar métodos de clustering:** Esta responsabilidad es la que se encarga de devolver todos los métodos de clustering (con sus parámetros). ✓ **Ejecutar métodos de clustering:** Esta responsabilidad es la que se encarga de recibir los datos de entrada para la ejecución asíncrona del algoritmo de clustering seleccionado. ✓ **Validar el procesamiento de los algoritmos de clustering:** Esta responsabilidad es la que se encarga de validar que el procesamiento asíncrono haya terminado. ✓ **Retornar los resultados:** Esta responsabilidad es la que se encarga de devolver los resultados una vez ejecutado el algoritmo de clustering seleccionado.
Clustering Web App	Este componente representa la interfaz web con la que el usuario va a interactuar. Estará desarrollada bajo el framework Angular 5 +. Debe permitir: ✓ **Permitir al usuario cargar datos:** Esta responsabilidad permite que el usuario pueda cargar un archivo .csv con los datos en los cuales se ejecutará el algoritmo de clustering. ✓ **Seleccionar el tipo de algoritmo de clustering:** Esta responsabilidad permitirá al usuario seleccionar un algoritmo de clustering y así mismo, indicar sus respectivos parámetros. ✓ **Consultar ejecuciones previas:** Esta responsabilidad permitirá consultar al usuario mediante un ID el resultado de una ejecución de clustering anterior.

Clustering Script Interfaz	Este sub componente debe permitir la ejecución de algoritmos de clustering creados en R desde los componentes creados en Java. Esto se realizará mediante el servidor Rserve. Debe permitir: ✓ **Establecer una conexión entre Java y RServe:** Esta responsabilidad debe crear una conexión para enviar con RServe todo lo concerniente a los algoritmos de clustering.
Clustering Scripts	Este sub componente es el responsable de implementar los algoritmos de clustering en scripts que serán desarrollados en el lenguaje de programación R. Debe permitir: ✓ **Recibir un data frame:** Esta responsabilidad debe establecer el data frame con los datos introducidos por el usuario o la API. ✓ **Serialización de respuesta a formato JSON:** Esta responsabilidad debe permitir que la respuesta otorgada o devuelta por el algoritmo de clustering sea serializada en formato JSON.
Clustering Data Base	Este componente debe permitir el acceso a la capa de datos definida en el documento de arquitectura de software. Debe permitir: ✓ **Recuperar la información de los algoritmos de clustering:** Esta responsabilidad debe acceder los datos relacionados a cada método de clustering dentro de la colección definida. ✓ **Recuperar y persistir los datos a analizar**

	mediante clustering: Esta responsabilidad debe permitir guardar la información (datos) que se envía para la ejecución de los algoritmos de clustering. ✓ **Recuperar y persistir los resultados del clustering:** Esta responsabilidad debe permitir recuperar o persistir los resultados arrojados por cada ejecución de clustering.

3.4.2 Relaciones de diseño

Las relaciones de diseño podrán apreciarse en la Sección No. 4 (Diagramas) en este documento, en los que se representan cada uno de los puntos de vista de diseño seleccionados.

3.4.3 Restricciones de diseño

En el presente proyecto no se contempla la aplicación de conceptos de seguridad y desempeño.

3.5 Plantillas de diseño

El diseño fue pensado en la reusabilidad y escalabilidad, por tanto el factor principal en su elección fue la arquitectura basada en microservicios.

3.6 Justificación del diseño

La finalidad del proyecto de software, de acuerdo a lo estipulado en el SRS, es la de desarrollar un componente que permita realizar análisis de clustering y que pueda ser usado por usuarios finales o integrarse con otras aplicaciones. Para ello, se pensó en la arquitectura de microservicios que permite tener soluciones escalables y desacoplar el funcionamiento de cualquier software en partes más pequeñas e independientes; lo cual a su vez permite que pueda usarse en múltiples entornos, desde ambientes ‘standalone’ hasta soluciones mucho más robustas.

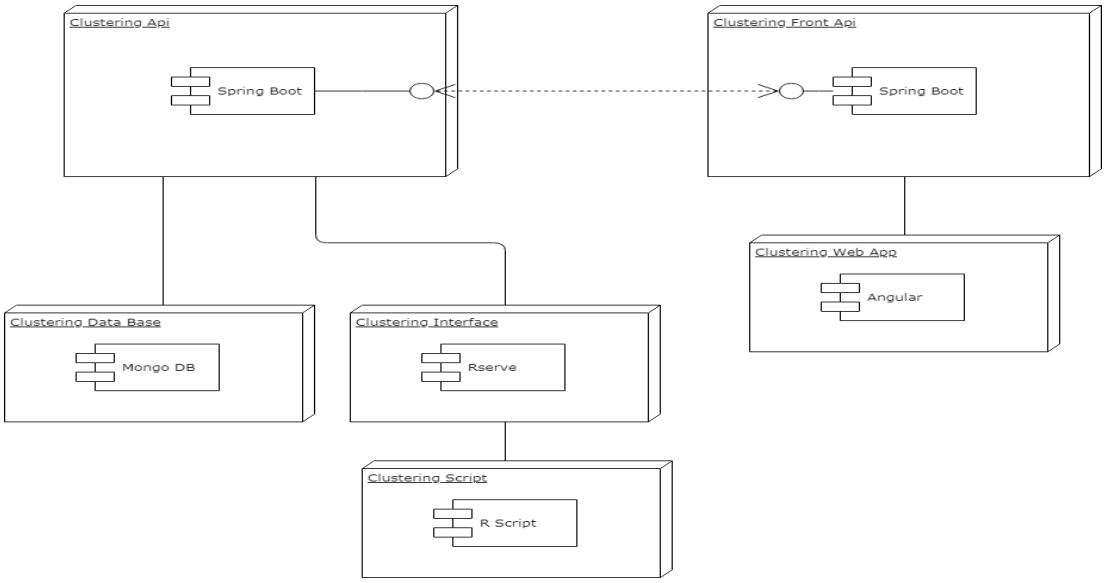
3.7 Lenguajes de diseño

Se utiliza para crear los puntos de vista el lenguaje UML.

4 Puntos de vista de Diseño (Diagramas)

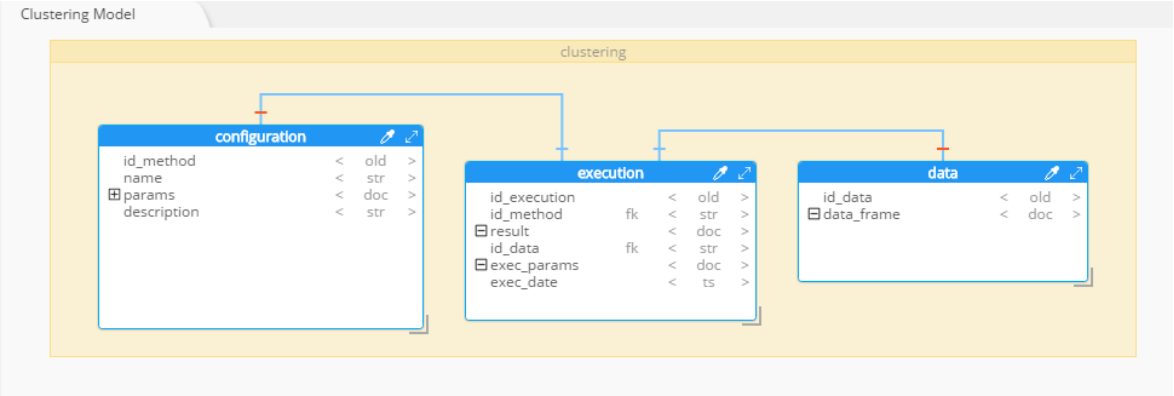
A continuación se representarán los puntos de vista y sus respectivas entidades, relaciones y restricciones de diseño.

4.4 Punto de vista de composición



Fuente: Joseph Rubio

4.5 Punto de vista de información



Fuente: Joseph Rubio

En el punto de vista de información, se determinaron tres colecciones:

1. La colección de configuración (configuration) donde se guardan la información de los métodos de clustering y sus parámetros, está conformada por:
    - Un ID único de cada método.
    - Nombre del método.
    - Parámetros de cada método.
    - Descripción del método.

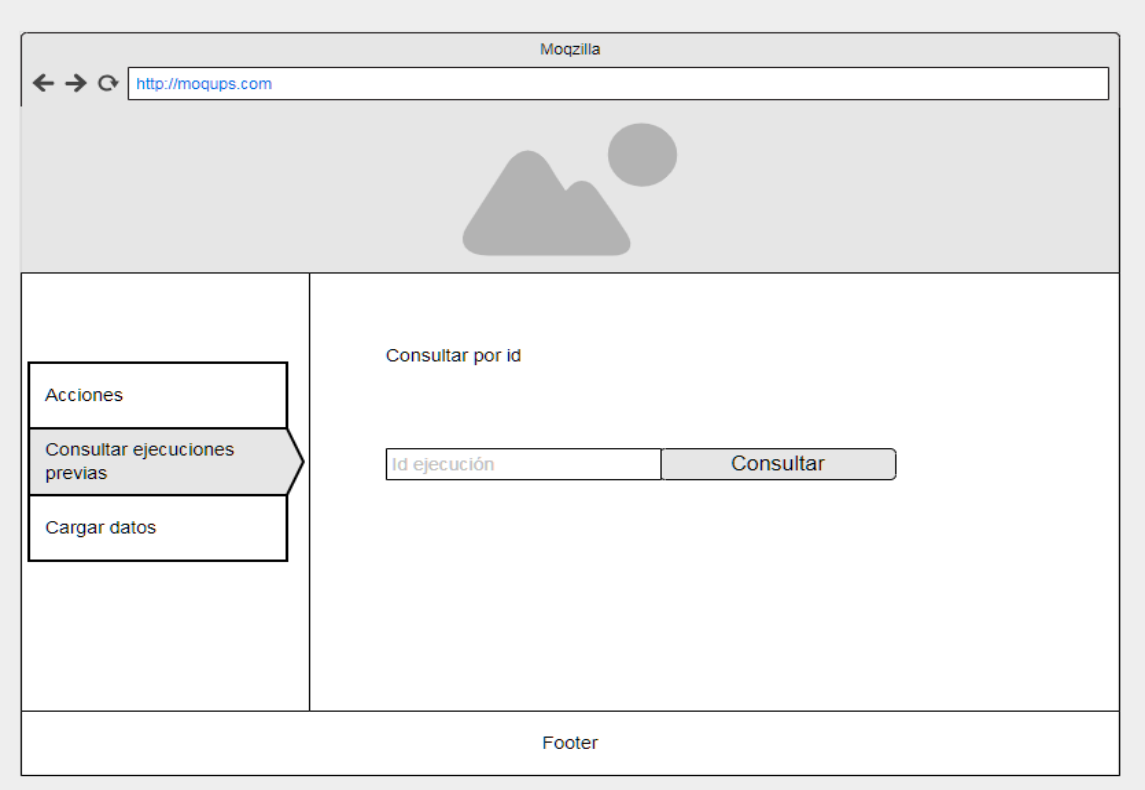
2. La colección de ejecución (execution) donde se guardara la información de las ejecuciones de los algoritmos de clustering, está conformada por:
    - Id de ejecución.
    - Id de método.
    - Resultados.
    - Id resultado.
    - Fecha de la ejecución

3. Colección datos (data), en ella se guarda el data frame que se analizara mediante clustering, está conformada por:
    - Id de los datos
    - Data frame
    - Fecha del cargue de datos

4.6 Punto de vista de interfaz

Se determinaros seis gráficos como vista de interfaz o mockups, los cuales todos tienen el mismo menú de opciones al lado izquierdo de la pantalla y constan de las acciones, consultar ejecuciones previas y cargar datos, dependiendo de la selección del usuario, la vista en la parte derecha se modificará.

Página Principal de Consulta



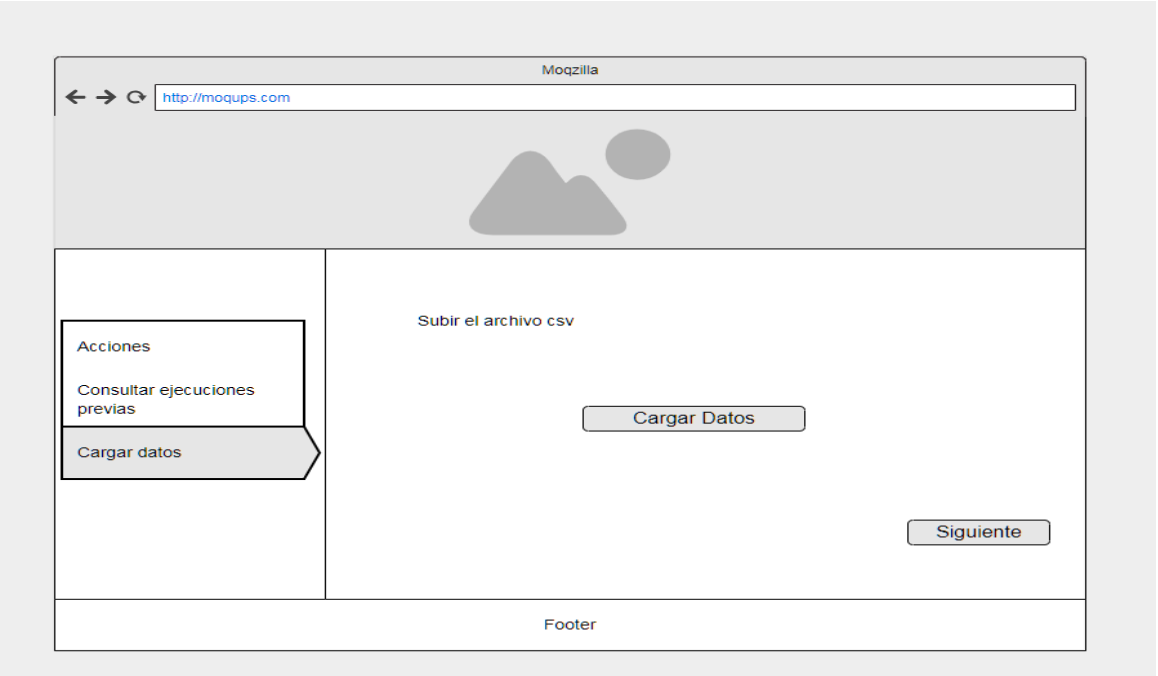
Fuente: Joseph Rubio

En la imagen denominada como **Página Principal de Consulta,** se evidencia que del menú izquierdo se seleccionó la opción ‘consultar ejecuciones previas’, por consiguiente, al lado derecho aparece un input, el cual sólo permitirá el ingreso de números con el que se consultarán resultados anteriores.

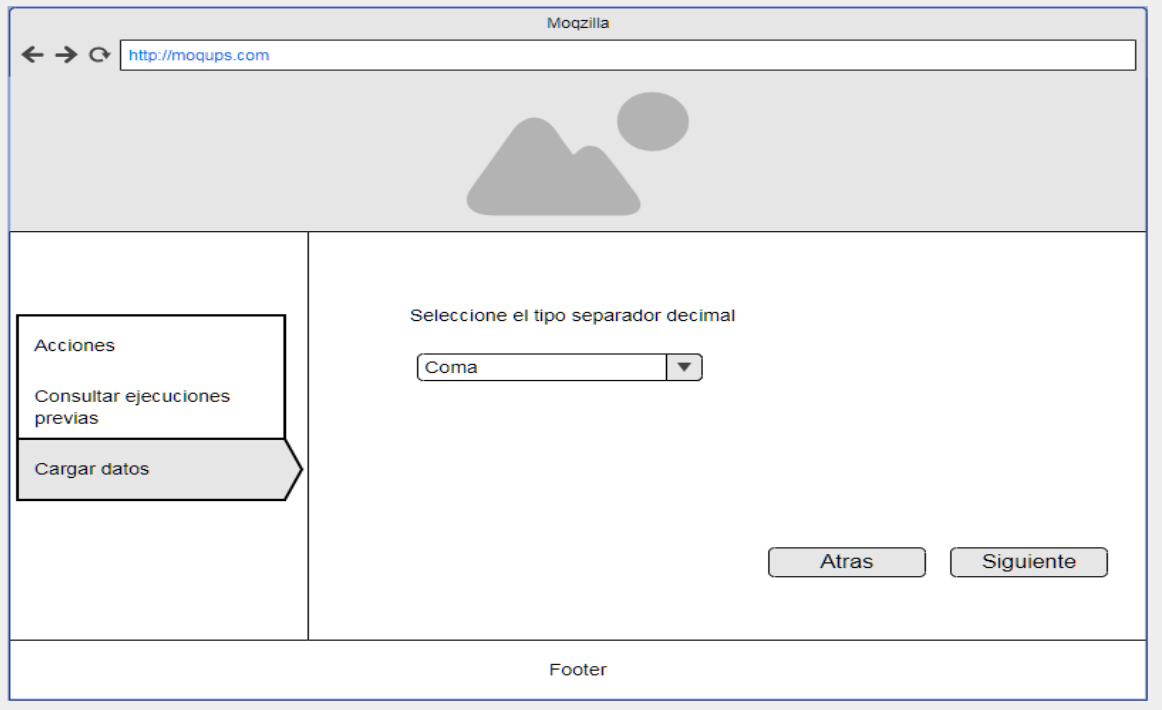
Página Principal Cargar Datos 1

A continuación, se muestra la transición de la vista según el proceso de cargar datos.

En la siguiente Imagen, se evidencia un mensaje donde se le informa al usuario que el documento que va a cargar debe ser en formato '.CSV', luego que el archivo esté adjunto se procederá a hacer clic en el botón "siguiente".

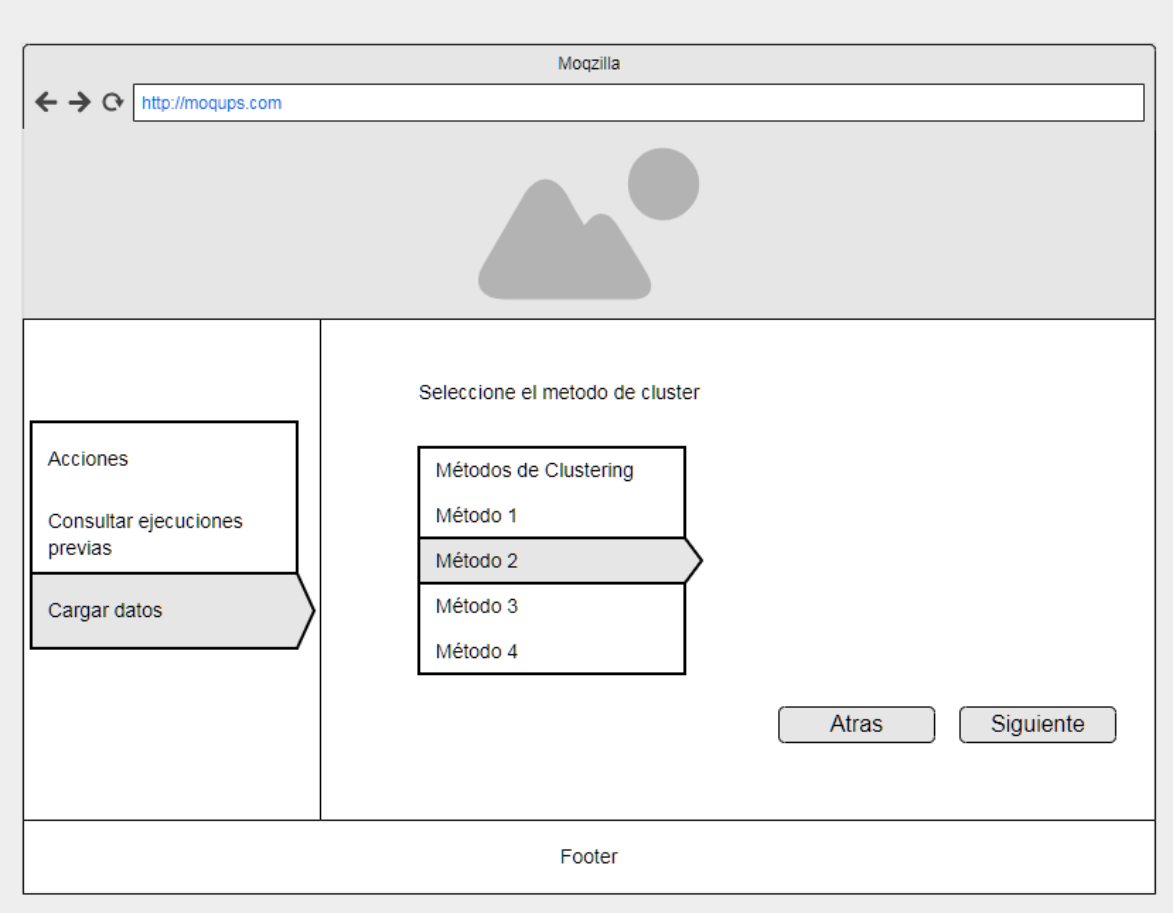


Fuente: Joseph Rubio



Fuente: Joseph Rubio

Luego se observará una lista que contiene los métodos de clustering disponibles para realizar el análisis.

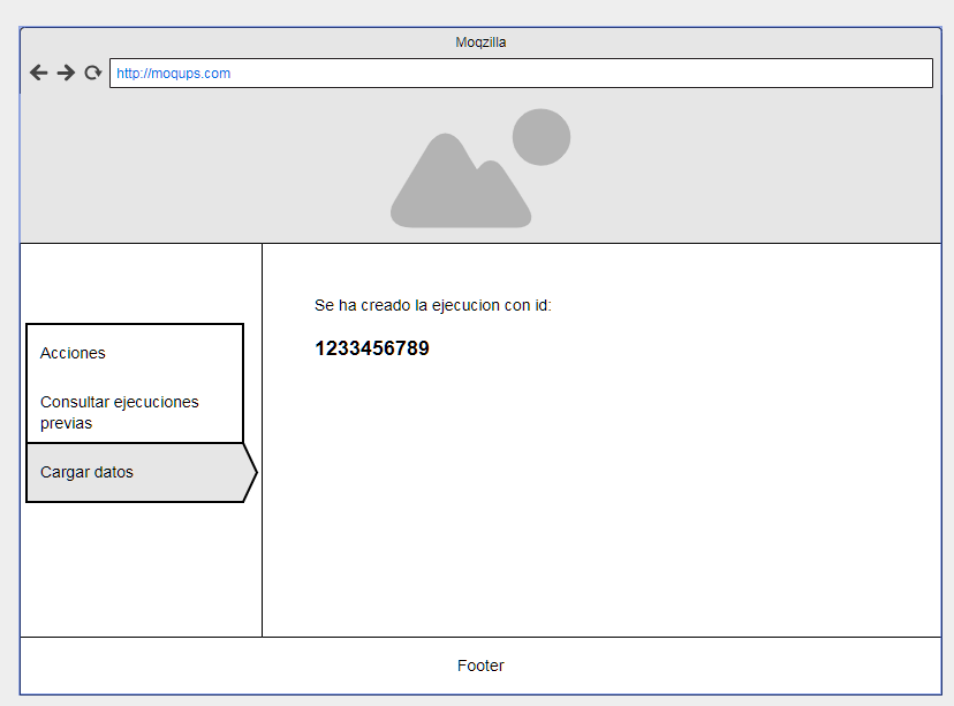


Fuente: Joseph Rubio

Una vez seleccionado el método de clustering, el componente web permitirá seleccionar los parámetros correspondientes y así proceder con la ejecución.

Fuente: Joseph Rubio

Luego que se realice la ejecución correctamente, aparecerá un mensaje informando el id del proceso.



Fuente: Joseph Rubi

ANEXO 3 (SAD)

DOCUMENTO DE ARQUITECTURA DE SOFTWARE (SAD)

COMPONENTE WEB PARAMETRIZABLE DE CLUSTERING PARA ANÁLISIS DE DATOS (CPWCAD)

JOSEPH ALEXANDER RUBIO TAPIAS

CARLOS ANDRÉS ALBA RODRÍGUEZ

DOCUMENTO DE ARQUITECTURA DE SOFTWARE

DIEGO ALBERTO RINCÓN YÁÑEZ MCSc

TABLA DE CONTENIDO

12. Control de Cambios

Fecha	Autor	Versión	Comentarios
28-10-2018	Joseph Rubio	1.0	Documento SAD.
19/11/2018	Joseph Rubio	2.0	Ajuste final

13. Introducción

Este documento define la Documentación de Arquitectura de Software (SAD por sus siglas en inglés), (Plantilla Estándar IEEE 1471-2000) (IEEE, 2012) para el desarrollo del Componente Web Parametrizable de Clustering para Análisis de Datos (CWPCAD), con el fin de definir las arquitecturas y presentar las vistas en los que dicho desarrollo se basa, de acuerdo al Documento de Especificación de Requerimientos de Software (SRS) presentado.

13.6 Propósito

Proporcionar una descripción de la arquitectura general del Componente Web Parametrizable de Clustering para Análisis de Datos, así como presentar una versión detallada, mediante la metodología 4+1 (Software & Kruchten, 2006), que involucra las diferentes perspectivas de la arquitectura.

1.2 Alcance

El alcance de este documento consiste en abarcar los diferentes elementos involucrados en la arquitectura del Componente Web Parametrizable de Clustering para Análisis de Datos, dichos elementos son:

- **Stakeholders**: Representan los actores involucrados en el desarrollo del componente, así como sus roles.

- **Vistas de Arquitectura**: Son los modelos representativos que describen la arquitectura general del componente.

- **Descripción de los Modelos**: Cada modelo consta de componentes que describen la arquitectura de acuerdo con las relaciones, dichos componentes tienen responsabilidades las cuales se han de describir.

Lo anterior, con el fin de tener una correcta documentación que apoye el diseño y el desarrollo del componente.

1.3 Usuarios Interesados

Principalmente, este documento va dirigido a todos aquellos usuarios que conocen del proyecto y que desean implementarlo. Adicionalmente, está construido con el fin de permitir que usuarios con conocimientos técnicos y nociones de desarrollo, tengan un marco de diseño en el cual basarse para realizar las fases de diseño, codificación y despliegue en el desarrollo del software.

2. Referencias

IEEE. (2012). *International Standard Iso / Iec Std 1471-2000* (Vol. 25021). https://doi.org/10.1109/IEEESTD.2015.7106438

Software, A., & Kruchten, P. (2006). Planos Arquitectonicos : El Modelo de “ 4 + 1 ” Vistas de la La Arquitectura L ´, *12*(6), 1–16.

3. Definiciones, Acrónimos y Abreviaciones

DAS: Documento de Arquitectura de Software

CWPCAD: Componente Web Parametrizable de Clustering para Análisis de Datos

SRS: Documento de especificación de requerimientos

4 Framework Conceptual

4.1 Descripción de la arquitectura en contexto

Para realizar la Arquitectura del CWPCAD se hará uso del Modelo ***de “4+1” Vistas de la Arquitectura de Software***, con el fin de poder describir y estructurar el comportamiento y diseño del mismo en cada una de las vistas que este modelo propone, dichas vistas son las siguientes:

- Vista de Escenarios
- Vista Lógica
- Vista de Proceso
- Vista de Desarrollo
- Vista Física

Adicionalmente, para el modelamiento de los diagramas que representan dichas vistas se hará uso del Lenguaje UML.

6.1 Stakeholders y sus roles

Los stakeholders y sus roles serán representados como aquellos que participan en el desarrollo del CPWCAD, ya que la idea de este es que pueda ser usado por

cualquier usuario, de tal manera que en la descripción de los stakeholders se hará referencia a cada uno como “Usuario final”, ya que, por el tipo de proyecto, no se cuenta con más usuarios involucrados, aparte de los que ya tienen rol en el ciclo de desarrollo del software.

6.2 Actividades de arquitectura en el ciclo de vida

N/A.

6.3 Usos de las descripciones de arquitectura

Los descriptores mencionados en este documento tienen como meta describir el diseño, estructura y comportamiento del CWPCAD, además de detallar la información necesaria para el desarrollo y comprensión por parte de los stakeholders.

7 Descripciones prácticas de arquitectura

5.1 Documentación de la arquitectura

N/A.

7.1 Identificación de los Stakeholders y sus responsabilidades

A continuación se describen los stakeholders involucrados en el CPWCAD:

Usuario Final: Es el stakeholder que interactúa con el CPWCAD a través de la interfaz prevista para el mismo y tiene como rol, además de operarlo, el poder realizar las pruebas funcionales. De acuerdo con el proyecto y como se definió anteriormente, el usuario final puede ser cualquier persona que interactúe con el CPWCAD.

Desarrollador: Es el stakeholder con mayor participación ya que interviene a lo largo de todo el ciclo de vida del desarrollo del software para el CPWCAD, por tanto, su rol es el de desarrollar y mantener.

Arquitecto de Software: Es el stakeholder involucrado en la fase de diseño y cuyo rol, además de dar la aceptación, es el de garantizar que el CPWCAD cumpla con lo estipulado en el SRS.

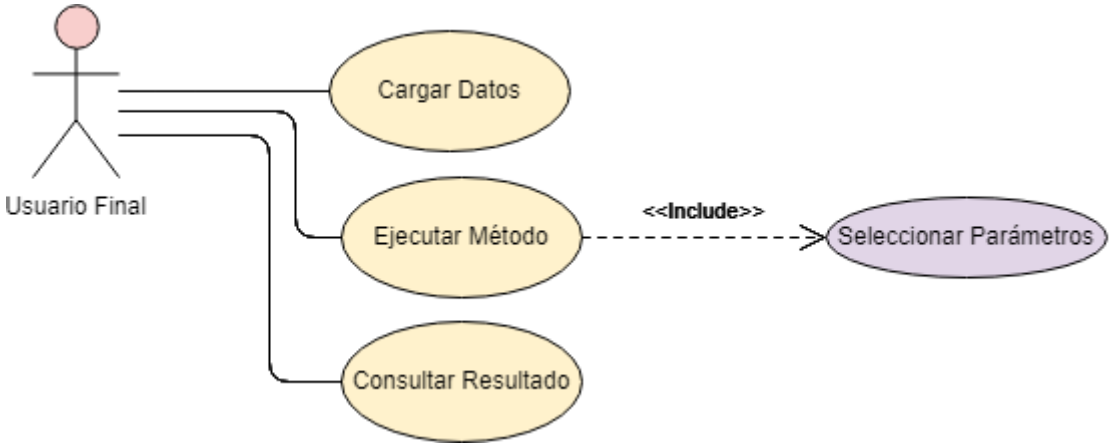
7.2 Selección de puntos de vista arquitectónicos.

A continuación, se seleccionan, siguiendo el modelo 4+1, los puntos de vista y su respectivo diagrama para la presentación de los mismos:

Punto de Vista	Diagrama UML
Vista de Escenarios	Diagrama casos de uso
Vista Lógica	Diagrama de paquetes
Vista de Proceso	Diagrama de secuencia
Vista de Desarrollo	Diagrama de componentes
Vista Física	Diagrama de despliegue

7.3 Vistas arquitectónicas

Casos de Uso - Usuario Final



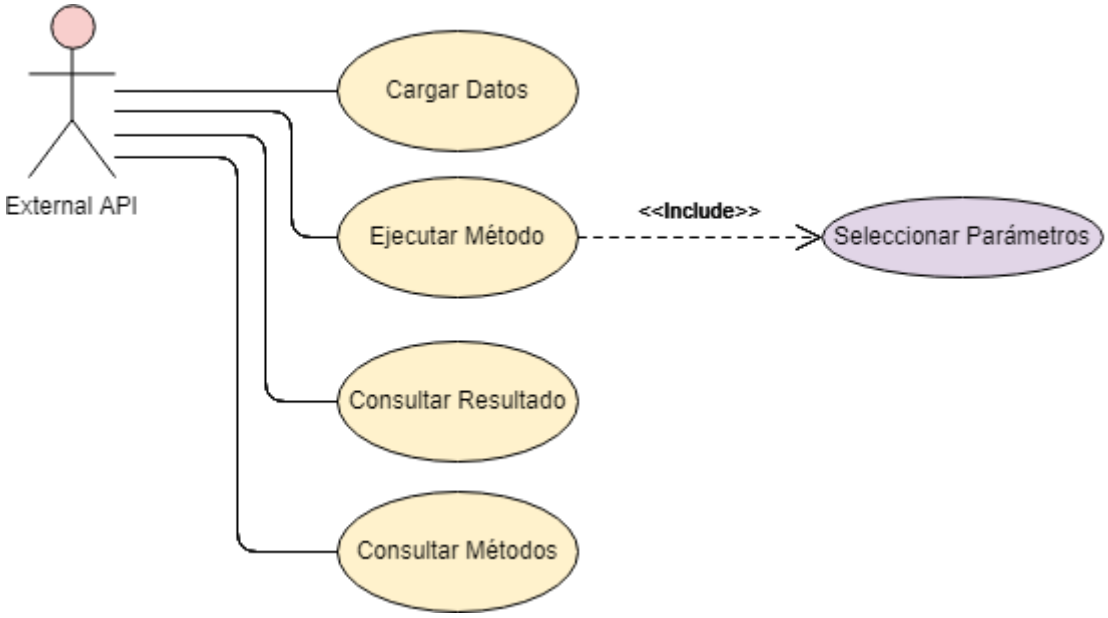
Fuente: Joseph Rubio

Descripción Caso de uso – Usuario Final

Caso de Uso	CU.1 Usuario Final
Actor	Usuario Final
Descripción	Se describen las acciones que puede realizar el usuario frente al componente web
Flujo básico	**1. Cargar datos**

	El usuario puede cargar un archivo .CSV con los datos que desea analizar.
	2. Seleccionar parámetros
	El usuario debe seleccionar los criterios adecuados para poder ejecutar los diferentes métodos de clustering, punto determinante para proceder a la ejecución.
	3. Ejecutar Método
	El usuario final debe indicar que desea iniciar con el proceso de clustering.
	4. Consultar resultado
	El usuario podrá visualizar los resultados obtenidos después de todo el proceso.
Pre-condiciones	**1. Archivo .CSV**
	El usuario necesita que los datos se encuentren en una estructura definida y en formato .CSV (comma-separated values)

Casos de Uso - API Externa

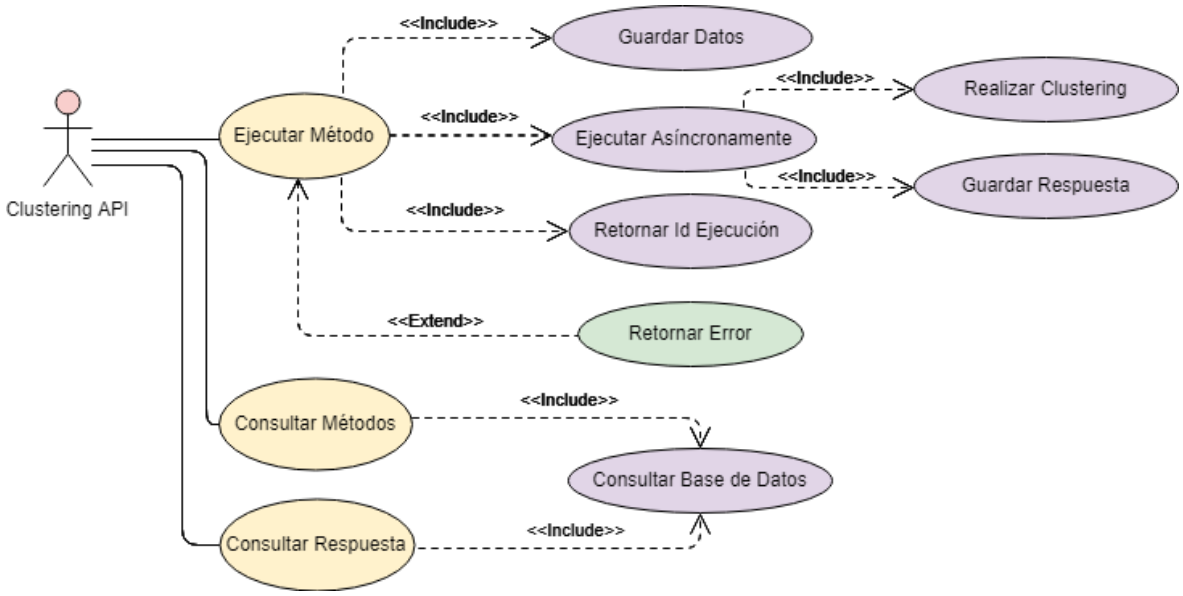


Fuente: Joseph Rubio

Descripción Caso de uso – API Externa

Caso de Uso	CU.2 API Externa
Actor	External API
Descripción	Las acciones del External API en gran medida se asemejan al usuario final, pero los datos que se ejecutan se encuentran en formato JSON, adicional a esto, el External API automáticamente consulta los métodos de clustering disponibles.
Flujo básico	**1. Cargar datos**
	El External API consulta datos en formato JSON
	2. Seleccionar parámetros
	El External API selecciona los criterios adecuados para poder ejecutar los diferentes métodos de clustering, punto determinante para proceder a la ejecución.
	3. Ejecutar Método
	Lleva a cabo el método de clustering según los parámetros ya seleccionados.
	4. Consultar resultado
	Los resultados quedan disponibles para su uso.
	5. Consultar Métodos.
	Verifica los métodos de clustering disponibles a ejecutar.
Precondiciones	**1. Datos en la base de datos**
	Se debe guardar con anterioridad en la base de datos los métodos de clustering que se pueden usar y sus respectivos parámetros.

Casos de Uso - API Externa

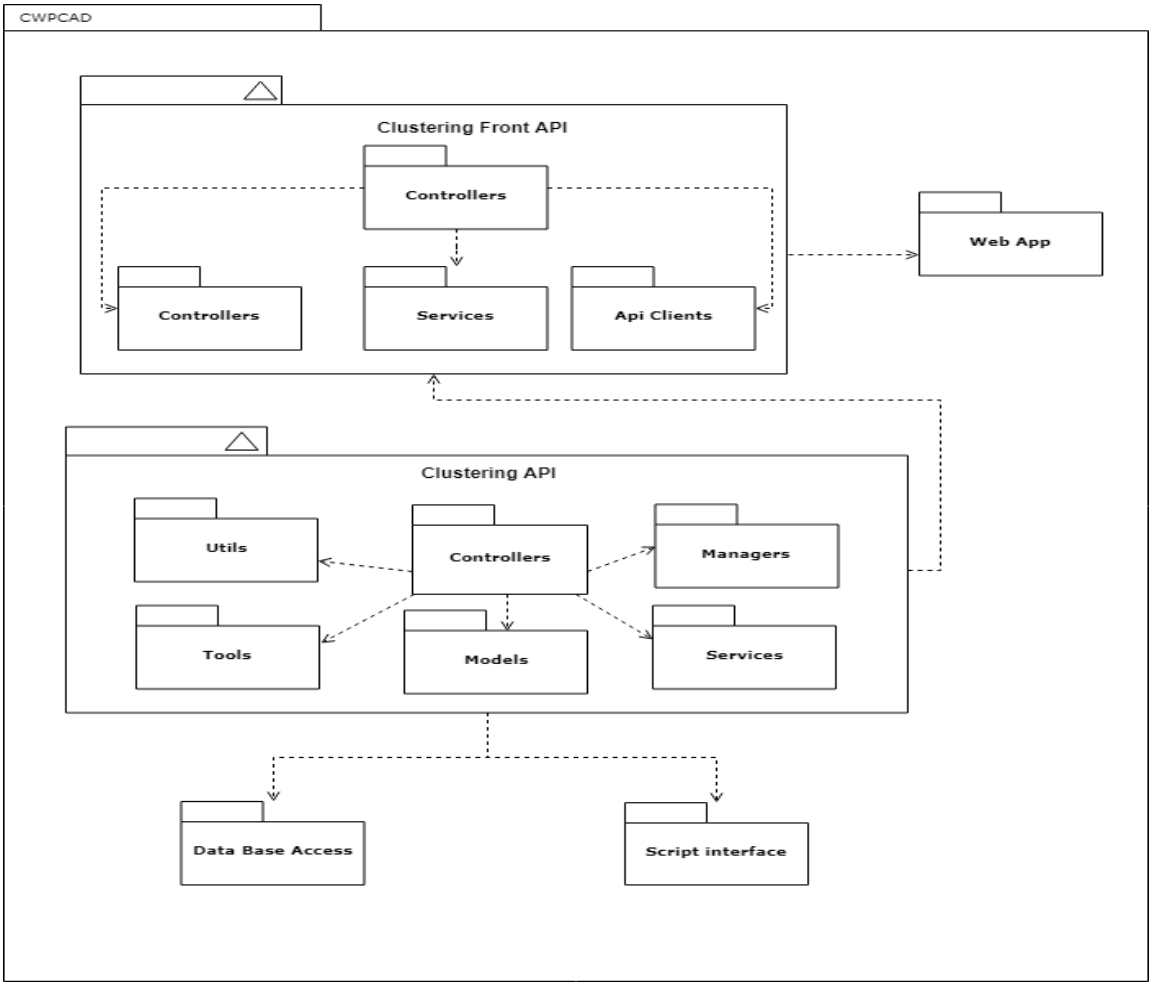


Fuente: Joseph Rubio

Caso de Uso	CU.3 API Clustering API
Actor	Clustering API
Descripción	Este caso de uso hace referencia de manera general al funcionamiento interno del componente web.
Flujo básico	**1. Ejecutar Métodos**
	Cuando el componente web inicia con la ejecución del método, internamente se están llevando a cabo las siguientes acciones: • Lleva a cabo el método de clustering. • Después del primer paso, el Clustering API guarda la respuesta en la base de datos. • Retorna el ID de ejecución. • En caso de que algo falle, retorna error.
	2. Consultar Métodos
	El Clustering API consulta en la base de datos los métodos disponibles y los parámetros asociados a cada uno.
	3. Consultar Respuesta

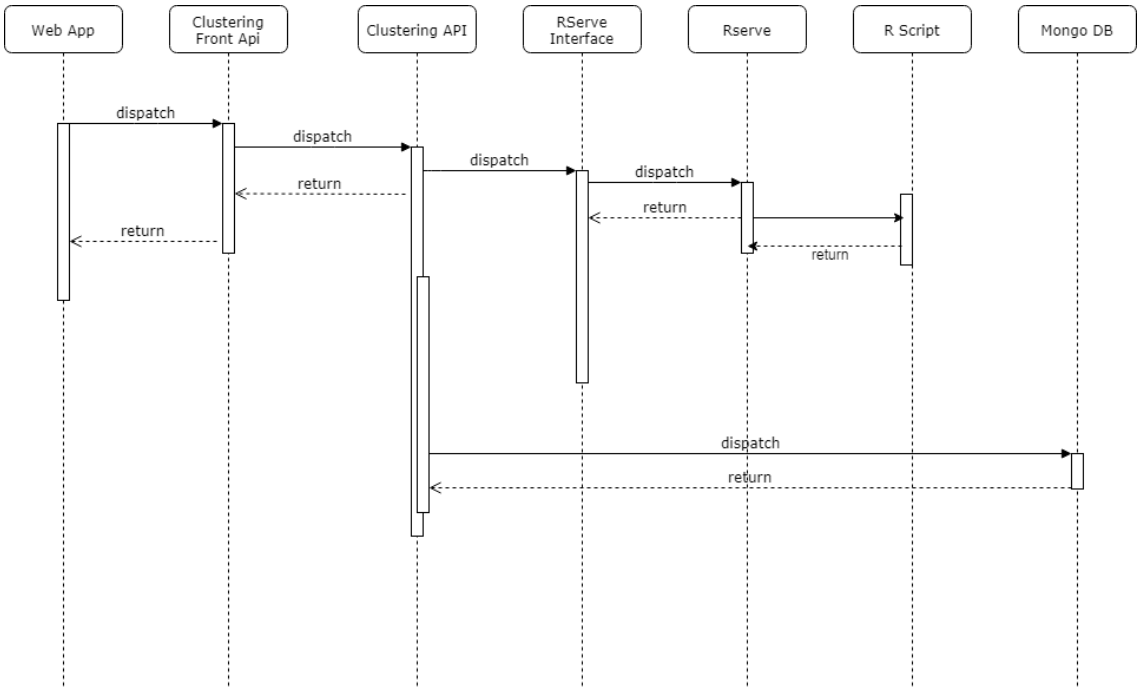
	El clustering API consulta en la base de datos los resultados arrojados en ejecuciones pasadas.

Vista Lógica



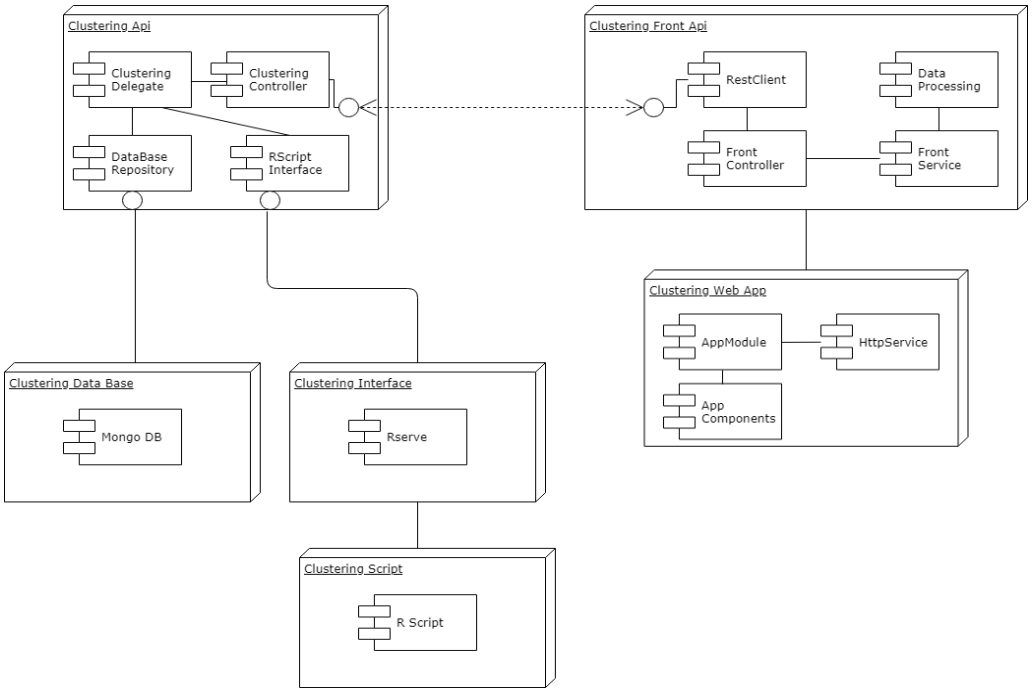
Fuente: Joseph Rubio

Vista de Proceso



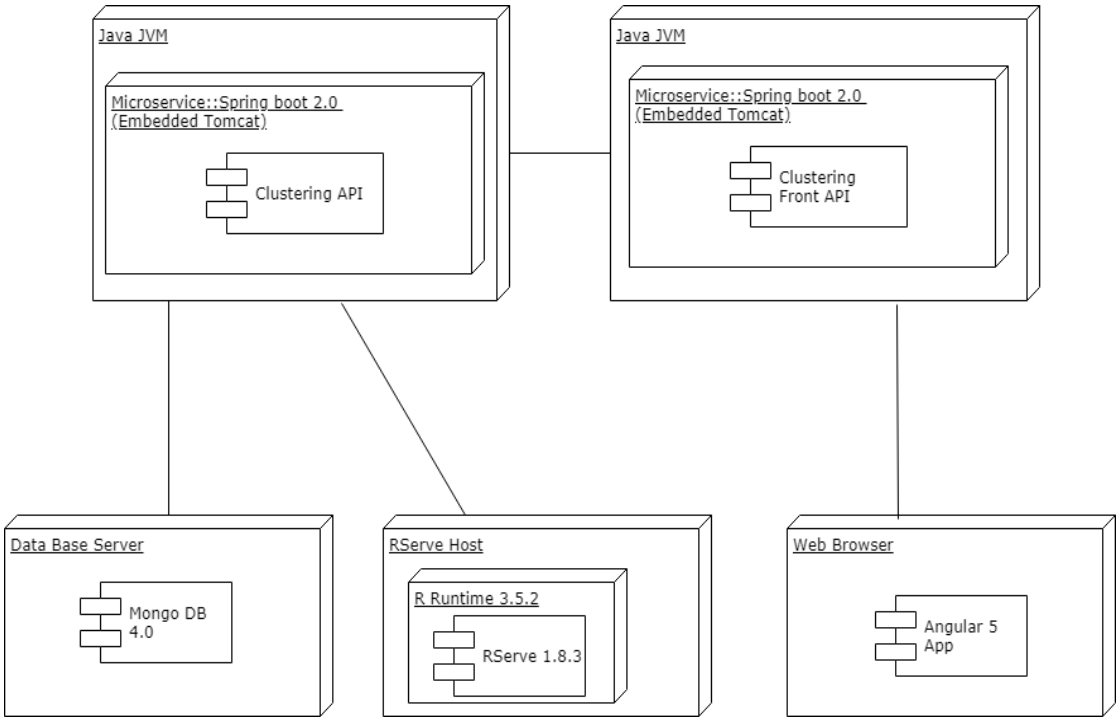
Fuente: Joseph Rubio

Vista de Desarrollo



Fuente: Joseph Rubio

Vista Física



Fuente: Joseph Rubio

7.4 Consistencia entre vistas arquitectónicas

A continuación, se explican cada uno de los módulos (nodos principales) y componentes representados en las diferentes vistas:

MÓDULOS O NODOS

WebApp: Representa la aplicación web del CPWCAD que tendrá interacción con el usuario final.

Clustering API: Representa la API con la cual se encuentra toda la lógica del CPWCAD, además de exponer los métodos con los cuales las aplicaciones externas pueden interactuar.

Front Clustering API: Representa la API que usa la **WebApp**, además de consumir al **Clustering API**.

Script interface: Representa la interfaz que permite al **Clustering API** utilizar los scripts.

Data Base Access: Representa la interfaz con el modelo de datos que el CPWCAD usará, de acuerdo con lo estipulado en el SRS.

Script: Representa el script de clustering para ser usado por el CPWCAD.

COMPONENTES

API Controller: Representa los controladores encargados de recibir las peticiones de procesamiento.

Data Base Manager: Representa el artefacto de software encargado de gestionar las operaciones de base de datos.

Pi Client: Representa el artefacto encargado de comunicarse con las API externas.

Script Connection: Representa el artefacto encargado de gestionar la conexión con el módulo encargado de ejecutar los scripts.

7.5 Justificación arquitectónica

La arquitectura representada en las vistas anteriores cumple con lo estipulado en el documento SRS, además de soportar a cabalidad la arquitectura de microservicios, cuyo fin es desacoplar en servicios más pequeños, reusables y escalables para funcionamiento y despliegue del CPWCAD.

MBRE TUT	TITULO DEL TRABAJO DE GRADO	CÓDIGO ESTUDIANTE	NOMBRES Y APELLIDOS ESTUDIANTE	OBJETIVO GENERAL	OBJETIVOS ESPECIFICOS	JURADO 1	JURADO 2
Diego Rincón	PARAMETRIZACIÓN DE SERVICIOS EN ALTA DISPONIBILIDAD PARA ECOSISTEMAS DE BIG DATA, UTILIZANDO HERRAMIENTAS DE AUTOGESTIÓN.	625014	Ricardo Fetecua Puentes	disponibilidad aplicados a ecosistemas de Big Data utilizando una herramienta de autogestión y autoconfiguración.	herramientas para servicios en alta disponibilidad inmersos en ecosistemas de Big Data. • Seleccionar 4 servicios en alta	Jenny Natalia Torres	Juan Carlos Barrero
Erika Holguín	APLICACIÓN DEL APRENDIZAJE AUTOMÁTICO EN LA CLASIFICACIÓN DE TEXTOS CORTOS: UN CASO DE ESTUDIO EN EL CONFLICTO ARMADO COLOMBIANO	625373	Cesar Augusto Espitia Betancourt	Clasificar textos cortos (tweets), extraídos de la red social twitter, en una orientación guerrerista, pacifista o neutra para un caso de estudio en	1. Realizar una revisión sistemática de la literatura para estructurar un estado de la cuestión relacionado con algoritmos de clasificación.	Raúl Bareño	Diego Alberto Rincón
		625381	Juan Pablo Paramo Lozada				
Diego Rincón	COMPONENTE WEB PARAMETRIZABLE DE CLUSTERIZACIÓN PARA ANALISIS DE DATOS NO ESTRUCTURADOS EN REDES SOCIALES	625167	Joseph Alexander Rubio Tapias	Desarrollar un componente de clusterización parametrizable para el análisis de datos de redes sociales	• Realizar un estado del arte de técnicas asociadas a la clusterizacion que puedan ser usadas en el análisis de datos de redes sociales. • Realizar el análisis de las	Yolandis Vega John Alexander	Nixon Alfonso Duarte
		625178	Carlos Andres Alba Rodriguez				

UNIVERSIDAD CATÓLICA de Colombia

caalba78 caalba78 <caalba78@ucatolica.edu.co>

Fwd: Incluir anteproyectos aprobados acta 24/10/2018

1 mensaje

Diego Alberto Rincón Yáñez <darincon@ucatolica.edu.co> 27 de octubre de 2018, 20:20
Para: caalba78 caalba78 <caalba78@ucatolica.edu.co>, Joseph Rubio <jarubio67@ucatolica.edu.co>, Ricardo Fetecua Puentes <rfetecua14@ucatolica.edu.co>

FYI

--

Diego Alberto Rincón Yáñez MCSc
Profesor Tiempo Completo
Departamento de Ingeniería de Sistemas y Computación
darincon@ucatolica.edu.co
Diagonal 46 A # 15 B – 10, sede Claustro, Bloque O, Piso 4
Bogotá, Colombia.
www.ucatolica.edu.co

UNIVERSIDAD CATÓLICA de Colombia

---------- Forwarded message ---------
From: **John Velandia** <javelandia@ucatolica.edu.co>
Date: Sat, Oct 27, 2018 at 12:47 PM
Subject: Incluir anteproyectos aprobados acta 24/10/2018
To: Holman <hdbolivar@ucatolica.edu.co>, HECTOR DARIO JAIMES PARADA <hdjaimes@ucatolica.edu.co>, INGENIERÍA DE SISTEMAS <sistemas@ucatolica.edu.co>, Trabajos de Grado - Ingeniería de Sistemas y Computación <ctg_sistemas@ucatolica.edu.co>
Cc: Diego Alberto Rincón Yáñez <darincon@ucatolica.edu.co>, <epholguin@ucatolica.edu.co>

Buenas tardes estimados

De acuerdo a la conversación que sostuvimos con Holman, por favor incluir en el acta de esta semana los proyectos que se adjuntan en el archivo de excel.

Confirmo que estos anteproyectos ya tuvieron el proceso de revisión por parte de los jurados y corrección de ajustes.

--

Many thanks/Muchas gracias/Vielen dank

Kind regards/Cordialmente/freundlichen Grüßen

John Velandia
Prof. M.Sc. Eng.
Facultad de Ingeniería - Programa Ingeniería de Sistemas y Computación
javelandia@ucatolica.edu.co
Tel. 3277300
Sede El Claustro - Diagonal 47, Cl. 47 #15-50, Bogotá - Colombia
www.ucatolica.edu.co

AVISO LEGAL:

- Las opiniones que contenga este mensaje son de su autor y no necesariamente representan la opinión **oficial** de la Universidad Católica de Colombia o de su directiva.
- El receptor deberá verificar posibles virus informáticos que tenga el correo o cualquier anexo a él, razón por la cual la Universidad Católica de Colombia no aceptará responsabilidad alguna por daños causados por cualquier virus transmitido en este correo.
- La información contenida en este mensaje y en los archivos electrónicos adjuntos es confidencial y reservada, conforme a lo previsto en la Constitución y en la Ley 1273 del 5 de Enero de 2009, y está dirigida exclusivamente a su destinatario, sin la intención de que sea revelada o divulgada a otras personas. El acceso al contenido de esta comunicación por cualquier otra persona diferente al destinatario no está autorizado por la Universidad Católica de Colombia y está sancionado de acuerdo con las normas legales aplicables.
- El que ilícitamente sustraiga, oculte, extravíe, destruya, intercepte, controle o impida esta comunicación, antes de que llegue a su

Proyectos aprobados 2018-3.xlsx
11K